Hyper-parameter and Kernel Optimization in Support Vector Machines

by

Shinichi Yamada

A thesis
submitted to the University of Canterbury
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

University of Canterbury
2018

Abstract

Support vector machines (SVMs) are capable of producing high quality solutions for many types of real-world classification problems. SVMs classify the data by selecting hyperplanes that provide the maximum margin between classes. Intuitively, the farther a data point from the decision boundary (hyperplane), the lower the chances of making a false prediction. A fundamental theorem of SVMs proves that the generalization error rates of SVMs are independent of the dimensionality of the input space and they only depend on the ratio of the margin and the radius of spheres which contain all the data. The expressive power of SVMs can be extended by mapping input data into high dimensional spaces. In the high dimensional spaces, the computational complexity of SVMs does not increase much, because all of the computation is done through the kernels—"the kernel trick". The maximal margin principle and the kernel trick make SVMs useful and efficient computational tools in machine learning.

While the parameters (support vectors) of SVMs can be found efficiently, the choice of kernels and optimal values of hyper-parameters within the kernels remains an open problem. This thesis addresses the issues of finding the optimal hyper-parameters and constructing the optimal kernel. Since the search spaces of these problems are very large, and often lack properties such as convexity that can be exploited, the thesis tackles the problem by proposing a set of tools that are based on (or influenced by) meta-heuristic algorithms and evolutionary computation.

For the hyper-parameter optimization, we develop a surface estimation method using the Bézier curve method. Firstly we determine the search region and choose samples on a regular grid in the region. Then we construct a surface which interpolates the prediction accuracy on sampling points and detects the most relevant region for a finer second search. The proposed method makes the search process more efficient and automatic, and it can be applied to multi-dimensional spaces.

The optimization of hyper-parameters can be further improved by making the detection of the initial search region automatic and by making the search process within this region more efficient. We achieve this goal by taking an approach based on evolutionary computation. Particle Swarm Optimization (PSO) is an algorithm inspired by the social behaviour such as in a bird flock and fish school. We propose a unified

approach for discretization of continuous PSO. Currently binary PSO and continuous PSO are two separate subjects, and most of the theories developed for continuous PSO can not be applied to binary PSO. We propose a general model to combine these two subjects and derive several binary PSO models and discrete PSO models (where each dimension can have more than two values) from existing continuous PSO models.

We apply the proposed discrete PSO to the hyper-parameter search in SVMs. In order to further increase the efficiency, we introduce an adaptive calibration of the evaluation points. We also present a simple algorithm to automatically adjust the search region using PSO. In the experiments, we show that the proposed discrete PSO is more efficient than surface estimation methods.

For constructing the optimal kernel, we propose a new multiple kernel learning (MKL) method which minimizes the ratio of the radius and the margin rather than just maximizing the margin. The proposed method has several favorable properties. It does not need to tune additional parameters and returns high quality solutions comparable to the best solutions of traditional MKL methods with fine tuned parameters.

The application of MKL does not always improve the prediction accuracy of simple SVMs with fine-tuned parameters. To improve the prediction accuracy of SVMs, we construct optimal kernels from possible nonlinear combinations of kernels and subsets of features. We propose a system of constructing optimal kernels using Genetic Programming and binary PSO. In the experiments, we show that the proposed method drastically improves the prediction accuracy over SVMs with fine-tuned parameters.

It is found that for both tasks of hyper-parameter optimization and optimal kernel construction, the proposed methods solve the difficulties imposed by standard methods in SVMs. This is achieved by combining the standard methods with meta-heuristic algorithms which improves the performance in terms of accuracy and efficiency.

Acknowledgments

I would like to express my gratitude to those who supported me during my PhD study.

My deepest gratitude goes to my supervisor, Dr Kourosh Neshatian. He let me explore various research directions. It took a long time but I finally found a few promising research ideas. He is strict with himself but very kind with others. This thesis would not have been possible without his guidance and support.

I am grateful to Dr. Lucianne Varn for reading and improving my articles.

I wish to thank Dr. Tadao Takaoka. I appreciate his wit and humor. Unfortunately he passed away in November 2017. He helped me to start my PhD research at University of Canterbury.

I would like to especially thank my wife and children for their love, support and patience.

Contents

Part I

Introduction

Chapter 1

Overview

In this chapter, we provide the overview of the thesis along with objectives and main contributions.

1.1 Problem Statement

In the 1950s, the Perceptron, introduced by Rosenblatt [130], was designed to solve classification tasks by separating data using hyperplanes. This inspirational work has had a big impact on the machine learning community and still lies at the centre of the research in online learning. Vapnik and Chervonenkis [161] introduced the "VC dimension" and opened up numerous possibilities for multi-variate analysis. They proved that generalization error rates can be controlled by the capacity of sets of functions regardless of the dimensionality of spaces. Vapnik and his colleagues also developed maximal-margin classification systems called Support Vector Machines (SVMs) (Cortes and Vapnik [29], Boser et al. [17]). SVMs separate instances of different classes using hyperplanes in the same way as the Perceptron, but the hyperplanes are chosen so as to maximize the margin between the classes. In the 1990s, two important modifications were added to the SVMs. The first one was the introduction of "slack" variables, which allow some miss-classifications due to the noise in data (Cortes and Vapnik [29]). It transformed the SVMs to a more general framework called regularized loss minimization, which consists of a loss function and a regularization term. The second one was the incorporation of nonlinear mapping, which was cleverly done via the inner products of nonlinear functions, called kernels (Boser et al. [17]).

In the framework of the regularized loss minimization, if the loss functions are convex, bounded and Lipschitz, then the optimization problems are "learnable" for

any distribution of data (Shalev-Shwartz and Ben-David [137]). Therefore, combined with the maximal margin principle and the computation via kernels, the optimization problems of SVMs are solved accurately and efficiently for any distribution of data. However, the choice of kernels and optimal values of hyper-parameters—a regularization hyper-parameter which balances between the loss function and the regularization term, and kernel hyper-parameters which are parameters in kernels—remains an open problem.

1.2 Weakness and Challenges in Current Research

Hyper-parameters must be specified before solving the optimization problem (learning). This means that when searching for optimal values of hyper-parameters, we need to solve the optimization problem for each specification of the hyper-parameters. A number of methods have been proposed to deal with finding optimal values for hyper-parameters. Chapelle et al. [24] proposed a search method based on the gradient descent. However, there is no guarantee that the algorithm arrives at the global optimal solutions. Keerthi and Lin [69] proposed a heuristic search method for the estimation of optimal hyper-parameters in Gaussian kernels. (This method explores the pairs of hyper-parameters $(\log \sigma^2, \log C)$ along a straight line with a unit slope which starts at the solution of a linear SVM obtained by $\sigma^2 \to \infty$.) The "path analysis" has been developed to quickly solve the optimization problems for each specification of hyper-parameters. Efron et al. [42] found that the optimal coefficients of "Least Absolute Shrinkage and Selection Operator" (LASSO) can be expressed by a piecewise linear function of a hyper-parameter. The same result was also extended to SVMs (Hastie et al. [62]). Later it turned out that the worst case complexity of the solution path of SVMs is exponential, in both the number of records and dimensions (Gärtner et al. [1]). Therefore, the main interest has shifted towards approximation methods of the path analysis rather than seeking the exact solutions (Giesen et al. [51–53]).

In spite of many efforts, the most common method for hyper-parameter optimization in SVMs is still the grid search. The reason being that it is simple to implement and that by setting the step size to a small valued the output of the algorithm approaches that of an exhaustive search (which guarantees finding optimal solutions). However, in grid search, the number of combinations of specific values of hyper-parameters grows exponentially with respect to the number of hyper-parameters. Therefore, even in three or four dimensional spaces, it is challenging to apply the grid search efficiently

and obtain reliable results.

Multiple kernel learning (MKL) was first introduced by Lanckriet et al. [85], who showed that when using a combination of kernels, the coefficients of the kernels and the optimal solutions of SVMs can be solved simultaneously. In MKL, once a set of kernels and a set of hyper-parameters in the kernels are prepared, the optimal coefficients are automatically computed and returned. Therefore, MKL can be used to evaluate multiple kernels with only one optimization run. This is a valuable property when searching for an optimal combination of kernels (kernel construction).

With regard to accuracy, MKL does not necessarily perform as well as SVM. One reason is that MKL loses the Lipschitz property in order to keep the convexity and the linear approximation is necessary to solve the MKL in the primal form. New theoretical developments are necessary in order to address these limitations. This is the subject of one of the chapters in this thesis.

Besides, the study of MKL has been mainly focused on the linear combination of kernels and the study of non-linear combination of kernels has been limited. However, different subsets of features, different set of kernel hyper-parameters, and the linear or nonlinear combinations of kernels, should all be considered when constructing kernels. One promising direction to tackle the problem of searching the vast space of kernels is to take an approach based on Evolutionary Computation (EC), which is a group of population-based global optimization algorithms and includes most of nature-inspired algorithms. Since the search spaces are very large and lack convenient properties such as convexity, the EC approach provides a natural and sophisticated means to tackle the problem (Riolo et al. [126], Koza [82], Bonabeau et al. [13]).

1.3 Goals

The overarching goal of this thesis is to find methods to improve the process of configuring SVMs. More specifically, the goals are to find better algorithms for:

- hyper-parameter optimization in SVMs; and
- optimal kernel construction in SVMs.

The thesis tackles the problems by proposing a set of tools that are based on a combination of methods directly related to SVMs and EC algorithms. The criteria for success in achieving these goals are improvements in accuracy and efficiency.

1.4 Major Contribution

The thesis has made the following major contributions towards its goals:

Hyper-parameter Optimization by Surface Estimation. Bézier curves are standard tools for curve and surface description in CAD/CAM and computer graphics. We construct a tool based on the Bézier curve method which interpolates the prediction accuracy on sample points in the search space. We compute the volume under the surface and identify the most important region for a finer second search. The surface estimation methods make the search process more efficient and automatic, and can be applied to multi-dimensional spaces. These results have been partly published in (Yamada, Neshatian and Sainudiin [174]).

Hyper-parameter Optimization by Discrete PSO with Dynamic Calibration. Particle swarm optimization (PSO) is a popular meta-heuristic method which produces computational intelligence through social interaction [73]. We experimentally verify that PSO is in general an excellent tool for hyper-parameter search in SVMs. To further enhance the efficiency of the PSO, we propose the discrete PSO which dynamically changes the calibration between evaluation points according to the density of particles. The proposed discrete PSO with the dynamic calibration improves the efficiency of the PSO without sacrificing accuracy (Yamada and Neshatian, [171]).

Weight-Radius Multiple Kernel Learning. We develop a new MKL method which minimizes the products of norms of weights of hyperplanes and the radius of the sphere which includes all the data. We prove that the proposed MKL is a closed system, and we can solve the optimization problems without imposing additional constraints. The proposed MKL returns high quality solutions comparable to the best ℓ_p-norm MKL with the fine tuned parameter p (Yamada and Neshatian, [173]).

Optimal kernel construction by Genetic Programming (GP). Genetic Programming (GP) evolves computer programs by applying genetic operations. To improve the prediction accuracy of SVMs, we develop a system that uses GP and binary PSO to explore the space of possible nonlinear combinations of kernels and subsets of features. We experimentally show that this approach can attain significant improvement of prediction accuracy when compared to the standard implementation of SVMs (Yamada and Neshatian, [172]).

The thesis has also made the following major contributions:

Discretization of continuous PSO. We propose a unified approach for discretization of continuous PSO. The contributions of this approach are as follows.

- We introduced a unified model in which binary and discrete PSO are variants of continuous PSO and the discrete versions of PSO are derived from the existing PSO. In this approach, the discrete PSO benefits from various inventions and theoretical achievements in the continuous PSO.
- We also presented global and local convergence theorems for discrete versions of PSO in the proposed framework.
- For the binary PSO we showed that the proposed "position as probability" approach is not only theoretically supported but also outperforms the commonly-used "velocity as probability" approach. As stated above we can expect a lot of improvements in this area because the binary PSO is now directly connected to the development of the continuous PSO.
- We demonstrated the capabilities of the new discrete PSO in optimizing combinatorial problems by applying it to Independent Job Scheduling Problems.
- We proposed Gray code as an appropriate coding scheme of bit strings for the experiments of binary PSO using benchmark functions. We verified the claim in the experiments.

1.5 Organization of the Thesis

In this section, we provide an outline of the structure of this research and a guide for navigating the thesis.

1.5.1 Outline

The thesis is divided into four parts: (I) contains the introduction and provides the background to the research problems addressed; (II) contains the first part of the major contributions of the thesis: hyper-parameter optimization; (III) contains the second part of the major contributions: optimal kernel construction; (IV) contains the conclusion of the thesis. The outline of the chapters in each part is presented next.

I - Introduction

Chapter 1: Overview. This chapter contains the overview of the thesis, including the goals, contributions, and organization of the thesis.

Chapter 2: Background. This chapter carries out a review of fundamental concepts and theorems in Machine Learning (ML) and Evolutionary Computation (EC). For ML the review covers the fundamental concepts of learnability, major learning rules and basic formulations of SVM. For EC it covers basic models and theorems of continuous PSO and binary PSO and the basic architecture of GA and GP. It also provides a review of Bézier Curve Methods and the basic properties of Gray code.

II - Hyper-parameter Search in SVMs

Chapter 3: Hyper-parameter Search using Bézier Curve Methods. This chapter proposes a new surface estimation method for hyper-parameter search in SVMs using the Bézier Curve Method. It constructs simple surfaces called cubic Hermite tensor product surfaces. It introduces "α-percent region" to identify the most important region and conducts a finer second search within the α-percent region. The experiments show that the optimal solutions of the proposed surface estimation method are in most cases close to the optimal solutions obtained by exhaustively searching the space. This method is also used as a benchmark method to test new hyper-parameter search methods in Chapter 5. A review of Bézier Curve Methods is provided in Section 2.8.

Chapter 4: Discretization of Continuous PSO. This chapter proposes a discretization of the continuous PSO in a unified way. It introduces discretization functions to combine the discrete version of PSO with the continuous PSO. It derives some models of binary (discrete) PSO by applying the discretization functions to the continuous PSO, and shows the superior performance of the proposed models in the experiments. The discrete PSO and the binary PSO are used in Chapters 5 and 7, respectively.

Chapter 5: Hyper-parameter Search using Discrete PSO. This chapter proposes the application of the discrete PSO for hyper-parameter search in SVMs. In order to improve the efficiency, it adopts the adaptive calibration of the discrete PSO. In the experiments it verifies that the proposed discrete PSO with the dynamic calibration improves the efficiency of the PSO without sacrificing accuracy.

III - Optimal Kernel Construction in SVMs

Chapter 6: Weight-Radius Multiple Kernel Learning (WR-MKL). This chapter proposes a new MKL method which minimizes the products of norms of weights of the hyperplanes and the radius of the sphere that contains all the data. It solves the optimization problems using block coordinate descent methods. It proves a theorem of convergence to the unique global optimal solution. It experimentally verifies the high performance of the proposed methods, which is comparable to the best ℓ_p-norm MKL with the fine-tuned parameter p. WR-MKL is used in Chapter 7 for optimal kernel construction.

Chapter 7: Optimal Kernel Construction. This chapter proposes a system to explore the space of possible nonlinear combinations of kernels and subsets of features. The system consists of a subsystem of feature subset selection and a subsystem of optimal kernel construction. The first subsystem uses binary PSO to select the best n subsets of features, and the second subsystem takes the output of the first subsystem and finds an optimal combination of kernels using GP. Experiments show that this approach attains significant improvement of prediction accuracy in comparison to the standard implementation of SVMs.

IV - Conclusions

Chapter 8: Conclusions. This chapter concludes the thesis by presenting chapter-wise conclusions and an overall conclusion with respect to the research questions. It also suggests possible future research directions.

1.5.2 Navigation

To provide better navigation, Table 1.1 presents dependencies among the chapters of this thesis.

1.6 Benchmark Datasets

To test the new proposed methods we use benchmark datasets from the LIBSVM website [23], and the UCI Machine Learning Repository [90]. We have chosen datasets with a middle size of instances. The smallest dataset is "German" which has 1000 instances. Table 1.2 lists the datasets for binary classification, in which the categories for binary classification are given in the column "Binary Categories". Table 1.3 lists the datasets

Table 1.1: The dependencies of the contents of the thesis

Chapter	Title	Depends on
3	Hyper-parameter Search using Bézier Curve Methods	Section 2.8
		Section 2.2
		Section 2.3.1
4	Discretization of Real-valued PSO	Section 2.4
		Section 2.5
5	Hyper-parameter Search using Discrete PSO	Chapter 3
		Chapter 4
6	Weight-Radius Multiple Kernel Learning	Section 2.3
7	Optimal Kernel Construction	Section 2.7
		Section 2.6
		Chapter 4
		Chapter 6

for multi-class classification. We adopt the stratified sampling method, in which each class in training sets, validation sets and test sets are selected in proportion to the ratio of the class in the whole dataset.

In the experiments, a dataset is divided into three parts: training, validation, and test datasets. The validation datasets are used to compute the prediction accuracy during the training phase and for determining the hyper-parameters, while the test datasets are used for the final evaluation.

Our strategy is to use larger sizes of validation datasets and test datasets in order to obtain reliable results for the comparison.

Table 1.2: Benchmark Datasets

Dataset	Instances	Features	Classes	(Training/Validation/Test) Training $= 100$	(Training/Validation/Test) Training $= 300$	Binary Categories
Banknote authentication	1372	5	2	(100/450/822)	(300/450/622)	
Default of credit card clients	30000	24	2	(100/1000/3000)	(300/1000/3000)	
Crowd-sourced Mapping Data Set	10845	28	6	(100/1000/3000)	(300/1000/3000)	$= 2$ vs. $\neq 2$
Sensorless Drive Diagnosis	58509	49	11	(100/1000/3000)	(300/1000/3000)	$<= 5$ vs > 5
German	1000	24	2	(100/400/500)	(300/300/400)	
Letter	15000	16	26	(100/1000/3000)	(300/1000/3000)	$<= 13$ vs > 13
MAGIC Gamma Telescope	19020	11	2	(100/1000/3000)	(300/1000/3000)	
Occupancy Detection	20560	7	2	(100/1000/3000)	(300/1000/3000)	
Optical Handwritten Digits	5620	62	10	(100/1000/3000)	(300/1000/3000)	odd vs even
Page Blocks Classification	5473	10	5	(100/1000/3000)	(300/1000/3000)	1 vs > 1
Pen-Based Handwritten Digits	10992	16	10	(100/1000/3000)	(300/1000/3000)	odd vs even
Satellite	6435	36	6	(100/1000/3000)	(300/1000/3000)	$<= 3$ vs > 3
Segment	2310	19	7	(100/1000/1210)	(300/1000/1010)	$<= 3$ vs > 3
Splice	3175	60	2	(100/1000/2075)	(300/1000/1875)	
Statlog (Shuttle)	58000	9	7	(100/1000/3000)	(300/1000/3000)	1 vs > 1
Svmguide1	7089	4	2	(100/1000/3000)	(300/1000/3000)	
Wine Quality	4898	12	7	(100/1000/3000)	(300/1000/3000)	$<= 5$ vs > 5

Table 1.3: Benchmark Datasets for Multi-class Classification

Dataset	Instances	Features	Classes	(Training/Validation/Test)
Crowd-sourced Mapping Data Set	10845	28	6	(300/1000/3000)
Sensorless Drive Diagnosis	58509	49	11	(300/1000/3000)
Letter	15000	16	26	(390(15×26)/2600(100×26)/5200(200×26))
Optical Handwritten Digits	5620	62	10	(200(20×10)/1000(100×10)/3000(300×10))
Page Blocks Classification	5473	10	5	(300/1000/3000)
Pen-Based Handwritten Digits	10992	16	10	(200(20×10)/1000(100×10)/3000(300×10))
Satellite	6435	36	6	(300/1000/3000)
Segment	2310	19	7	(300/700/1310)
Statlog (Shuttle)	58000	9	7	(300/1000/3000)
Wine Quality	4898	12	7	(300/1000/3000)

Chapter 2

Background

In this chapter, we present a review of machine learning and evolutionary computation. The thesis aims to develop some methods in these two fields.

2.1 Overview

This chapter starts with a review of machine learning. We provide an overview of machine learning and consider what the "learning" means.

In the next section we derive the formulation of Support Vector Machine (SVM) in a geometrical point of view and consider the maximal margin principle of SVM. We provide a review of kernels and two major implementation algorithms of SVM.

In the following sections, we review the fields in Evolutionary Computation (EC). We start with an overview of EC. Then we review basic concepts and models in Particle Swarm Optimization (PSO). Several standard continuous PSO and discrete (binary) PSO models are reviewed. Then the sections of Genetic Algorithm (GA) and Genetic Programming (GP) follow. We explain basic components and operations, and work flows of these methods.

In the next section we provide a review of Bézier Curve Methods. Bézier curve is a standard method to draw curves and surfaces in blueprints and computer graphics. We will use the Bézier curve methods to draw the surface of the prediction accuracy on hyper-parameter spaces in Chapter 3.

In the next section we present a brief review of the basics of Gray code. In Chapter 4 we use Gray code to transform the real solutions of benchmark functions to bit strings. Gray codes are minimal change orderings of bit strings in which successive strings differ by a single bit.

In the last section we provide a literature review for the hyper-parameter search and feature selection in the machine learning fields and the evolutionary computation fields.

2.2 Machine Learning (ML)

In this section, we review what "learning" means and how to formulate it in Machine Learning (ML). We start with an overview of machine learning. Then we review two major learning rules; Empirical Risk Minimization (ERM) and Regularized Loss Minimization (RLM).

2.2.1 Overview

Learning is a process which converts experience into knowledge (Shalev-Shwartz and Ben-David [137]). The process includes memorization and generalization. The memorization is a first step of learning. For example, rats which had been fed with poisonous foods can avoid the next poisonous meal by remembering its looks and smell. The generalization is a more abstract process. It is a process of extracting some underlying characteristics based on a given experience and applying them to solve unseen problems.

Arthur Samuel [134] coined the name *machine learning* as:

> Machine learning explores the study and construction of algorithms that can learn from and make prediction on data.

Machine learning has its roots in diverse fields such as cybernetics [167], information theory [138] , Perceptron [130] and statistical learning theory [160].

The fields of machine learning have been divided into subfields which deal with different types of learning tasks. The first classification is based on the interaction between learners and the environment.

- Supervised learning: Learners are provided with a pair of input vectors and correct output labels. For instance, let us consider a task of detecting spam emails. In the supervised learning, the learners are provided with a set of emails with the corresponding labels, spam or non-spam. The task of learners is to learn from them and predict unseen emails.

- Unsupervised learning: Learners are provided with input vectors with no labels. For example, in the problem of anomaly detection of emails, the learners receive a large set of emails. In this case the task of learners is to detect unusual emails.

- Reinforce learning: Reinforce learning is the intermediate setting between the supervised learning and the unsupervised learning. Learners are provided with input vectors with summarized information in the form of rewards and punishments. For example, in the context of playing a game against an opponent, the learners have an access to the information of positions that occur throughout the games and the information about who eventually win the game. However, the information which evaluates each position is not provided to the learners.

The second classification of the fields of machine learning is based on the types of output labels.

- Classification: In classification, the output labels are binary responses or some values from a finite set. For example, in the spam email problem, the inputs are a set of email messages and the outputs are "spam" and "not spam" labels.

- Regression: In regression, the output labels are continuous numbers. Examples of regression include the prediction of stock values on some economic indexes.

- Ranking: In ranking, the output labels are integers which imply the ranking based on some criterion. For example, web search engines return the ranking of web pages which are relevant to the search query.

In this thesis we focus on the classification problems under the supervised learning setting. We outline some common methods of machine learning.

Artificial Neural Networks

An Artificial Neural Network is a computing system inspired by the structure of neural network in brain. In the system a number of basic components (neurons) are connected each other. ANNs are described by a directed graph (V, E) and a weight function $w : E \Rightarrow \mathbb{R}$ over edges. A node in V corresponds to a neuron which is a basic computing unit. A simple scalar function σ is carried out on each neuron. Commonly-used scalar functions are the sign function $\sigma(x) = sign(x)$:

$$sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

and the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$.

A feed-forward neural network is an acyclic model of ANN which does not contain cycles. We also assume that the set of nodes V are organized as layers $V = \bigcup_{i=0}^{N} V_i$. Every edge in E connects a node in V_i and a node in V_{i+1}. V_0 is called an input layer. For an input vector $\mathbf{x} = (x_1, \ldots, x_n)$, V_0 consists of $n+1$ nodes $v_{o,i}, i = 1, \ldots, n+1$. The first n nodes simply output $x_i, i = 1, \ldots, n$ and the last $(n+1)$-th node outputs a constant 1. $V_1, \ldots, V_{N-1}$ are called hidden layers and V_N is an output layer (Figure 2.1). Let $v_{t,i}$ be the i-th neuron in the layer V_t, $a_{t,i}$ be the input to $v_{t,i}$ and $o_{t,i}$ be the output from $v_{t,i}$. The input $a_{t,i}$ is a weighted sum of the output of neurons in V_{t-1} which are connected to $v_{t,i}$:

$$a_{t,i} = \sum_{(v_{t-1,j}, v_{t,i}) \in E} w((v_{t-1,j}, v_{t,i}))o_{t-1,j},$$

and the output $o_{t,i}$ is $\sigma(a_{t,i})$.

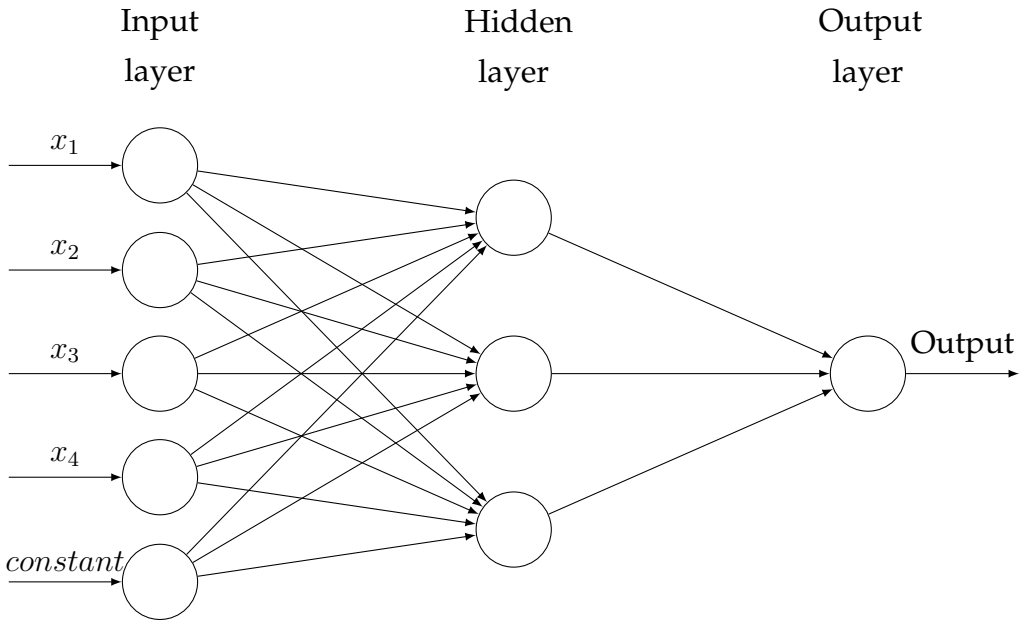


Figure 2.1: Artificial Neural Network

Let $f : \{\pm 1\}^k \to \{\pm 1\}$ be the conjunction function, $f(\mathbf{x}) = \cap_i x_i$. Then it can be written as $f(\mathbf{x}) = sign(1 - k + \sum_{i=1}^{k} x_i)$. The disjunction function $f(\mathbf{x}) = \cup_i x_i$ is also written as $f(\mathbf{x}) = sign(k - 1 + \sum_{i=1}^{k} x_i)$. Every boolean function $f : \{\pm 1\}^k \to \{\pm 1\}$ can be implemented by a neural network with depth 2, in which $|V_0| = n + 1$, $|V_1| = 2^n + 1$ and $|V_2| = 1$ (Shalev-Shwartz and Ben-David [137]). Therefore any conjunction and disjunction function can be implemented by a neural network with depth 2. If neurons

in the hidden layer implement a halfspace predictor $sign(\sum_{i=1}^{n} w_i x_i + w_{n+1})$ and the single neuron in the output layer implement the conjunction function, the network can express the intersection of those halfspaces (Figure 2.2). If we add one more layer and implement a disjunction function in the output layer, the network can express the union of polytopes (Figure 2.3). This rich expressive ability is a strong point of the artificial neural network.

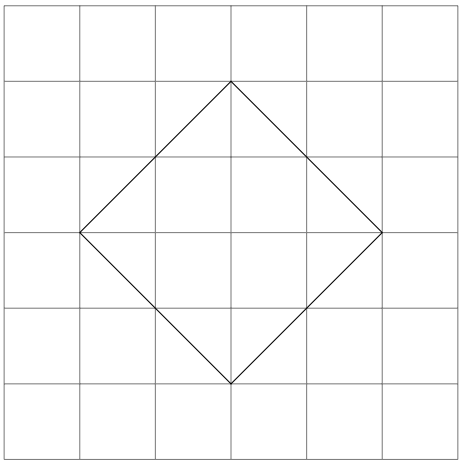


Figure 2.2: Diamond (Intersection of Halfspaces)

Support Vector Machines

Support Vector Machine (SVM) is a learning algorithm of linear predictors in high dimensional spaces. SVM classifies data using hyperplane in the same way as the Perceptron (Figure 2.4). It is rather a simple idea compared to Figure 2.3 of ANN. However, the classification is carried out in high dimensional spaces. In general, the sample complexity needed for the computation grows exponentially with the dimensionality of spaces. Therefore it becomes prohibitively large in high dimensional spaces. SVM tackles this problem by choosing hyperplanes which maximize the margin between classes. It is intuitively clear that the wider the distance between the point and the hyperplane the more confidence we have for the accuracy of the classification. The maximal margin principle keeps a small sample complexity even in high dimensional spaces. We will examine this principle in the section of SVM (Section 2.3).

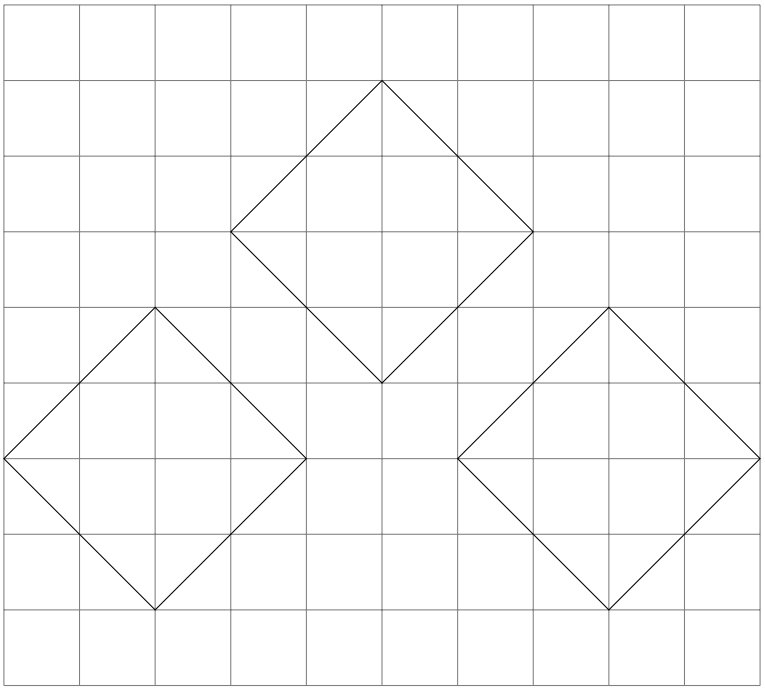


Figure 2.3: Union of Diamond

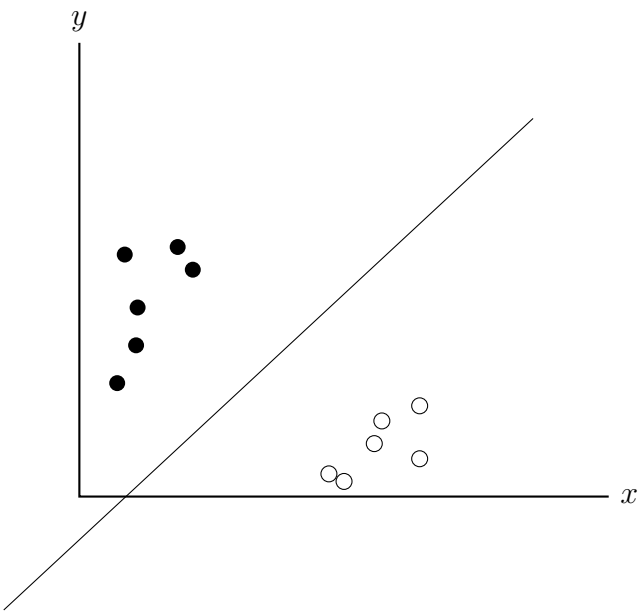


Figure 2.4: Classification by SVM

2.2.2 Empirical Risk Minimization (ERM)

In this section we review a basic learning rule, Empirical Risk Minimization (ERM). In order to construct mathematical models we start with some definitions with respect to data.

- Instances and features: Instances are elements in an arbitrary set $\mathcal{X}$. Instances are often represented as input vectors $(x_1, x_2, \ldots, x_d) \in \mathcal{X}$. The element (x_i) in an instance is called a feature.
- Labels: Labels are values and categories assigned to instances. Let $\mathcal{Y}$ denote a set of labels. In classification problems, instances are assigned specific categories, such as 0 or 1.
- Distribution: We assume that the instances are generated by some probability distribution $\mathcal{D}$ over $\mathcal{X}$.
- i.i.d assumption: The i.i.d assumption presumes that instances are independently and identically distributed according to a distribution $\mathcal{D}$.
- Training sets: A training set $\mathcal{S} \subset \mathcal{X} \times \mathcal{Y}$ is a sequence of labeled instances

  $$\mathcal{S} = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n) \ ,$$

  which is used to train a learning algorithm.
- Validation sets: A validation set is also a sequence of labeled instances which is used to tune the hyper-parameters in a learning algorithm. Learning algorithms typically contain some hyper-parameters. The validation sets are used to specify the optimal values of hyper-parameters.
- Test sets: A test set is a sequence of labeled instances which is separated from the training set and the validation set and is not made available in the training phase. The test set is used to evaluate the performance of the learning algorithm.
- Hypothesis: After the experiments with training sets, the main task of the learners is to derive a prediction rule $h : \mathcal{X} \Rightarrow \mathcal{Y}$, which is called a hypothesis. The hypothesis is used to predict unseen instances.

We also need to clarify some definitions with respect to the evaluation of learning process. Let $\mathcal{S}$ be a training set:

$$\mathcal{S} = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n) \ ,$$

where $\mathbf{x}_i = \{x_{i1}, \ldots, x_{id}\} \in \mathcal{X} \subset \mathbb{R}^d$ and $y_i \in \mathcal{Y} \subset \mathbb{R}$, are generated from a probability distribution $\mathcal{D}$.

We define an error of a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ as the probability that the event $h(\mathbf{x}) \neq y$ occurs:

$$Err_{\mathcal{D}}(h) := P_{(\mathbf{x},y)\sim\mathcal{D}}\left[h(\mathbf{x}) \neq y\right] \quad .$$

For a set $\mathcal{S}$, we define an empirical error of the hypothesis h as the ratio of the event $h(\mathbf{x}_i) \neq y_i$ in $\mathcal{S}$:

$$Err_{\mathcal{S}}(h) := \frac{|i \in \{1, \ldots, n\} : h(\mathbf{x}_i) \neq y_i|}{n} \quad .$$

To generalize the above notion of the error, we introduce a loss function $\ell\left(h(\mathbf{x}), y\right)$ which penalizes the deviation between the hypothesis $h(\mathbf{x})$ and y. For instance, a $0-1$ loss function and a square loss function (used in regression) are defined as

$$\ell_{0-1}\left(h(\mathbf{x}), y\right) = \begin{cases} 0 & \text{if } h(\mathbf{x}) = y \\ 1 & \text{otherwise} \end{cases} ,$$

and

$$\ell_{sq}\left(h(\mathbf{x}), y\right) = \left(h(\mathbf{x}) - y\right)^2 \quad .$$

We define a risk (a true risk, a generalization error) to be the expectation of the loss function,

$$L_{\mathcal{D}}(h) := \mathbb{E}\left[\ell\left(h(\mathbf{x}), y\right)\right]$$

and the empirical risk for the hypothesis h as

$$L_{\mathcal{S}}(h) := \frac{1}{n}\sum_{i=1}^{n} \ell\left(h(\mathbf{x}_i), y_i\right) \quad .$$

Since we do not know the underlying distribution $\mathcal{D}$, one way to estimate the optimal hypothesis is to seek for the hypothesis which minimizes the empirical risk:

$$h_{\mathcal{S}} \in \operatorname{argmin}_{h\in\mathcal{H}} L_{\mathcal{S}}(h)$$

where $\mathcal{H}$ is a set of hypotheses. This learning rule is called an Empirical Risk Minimization (ERM) rule.

Since the data in $\mathcal{S}$ is sampled from the distribution $\mathcal{D}$, $h_{\mathcal{S}}$ would be a reasonable guess for the optimal hypothesis. However, the ERM rule may fail when $\mathcal{H}$ contains "too elaborate" hypothesis. For instance, in a problem of fitting regression curve, a polynomial curve with degree $= 10$ may fit the training set better than a polynomial curve with degree $= 3$. However, it does not necessary generate better estimation for unseen data. This phenomena is called "over-fitting". When over-fitting occurs the ERM rule fails to learn the optimal hypothesis.

2.2.3 Regularized Loss Minimization (RLM)

In this section we introduce an another learning rule which prevents the over-fitting of convex, Lipschitz and bounded functions. We start with the definition of convex sets and convex functions.

Definition 2.1. *Convex Set*
A set $\mathcal{C}$ is convex, if for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{C}$, and for any $\alpha \in [0,1]$, the following holds:

$$\alpha \mathbf{w}_1 + (1-\alpha)\mathbf{w}_2 \in \mathcal{C} \ .$$

Definition 2.2. *Convex Function*
Let $\mathcal{C}$ be a convex set. A function $f : \mathcal{C} \mapsto \mathbb{R}$ is convex, if for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{C}$, and for any $\alpha \in [0,1]$,

$$f(\alpha \mathbf{w}_1 + (1-\alpha)\mathbf{w}_2) \leq \alpha f(\mathbf{w}_1) + (1-\alpha) f(\mathbf{w}_2) \tag{2.1}$$

Convex functions have the following property. This property is also used as an another definition of convex functions.

Lemma 2.1. *Let $\mathcal{C}$ be an open convex set. A function $f : \mathcal{C} \mapsto \mathbb{R}$ is convex, if and only if for every $\mathbf{w} \in \mathcal{C}$, there exists $\mathbf{v} \in \mathcal{C}$ such that*

$$\forall \mathbf{u} \in \mathcal{C}, f(\mathbf{u}) \geq f(\mathbf{w}) + \langle \mathbf{u} - \mathbf{w}, \mathbf{v} \rangle \tag{2.2}$$

Definition 2.3. *Sub-gradients (Shalev-Shwartz and Ben-David [137])*
A vector $\mathbf{v}$ which satisfies (2.2) is called a sub-gradient of f at $\mathbf{w}$. The set of sub-gradients of f at $\mathbf{w}$ is called the sub-differential set and denoted $\partial f(\mathbf{w})$. If a function f has a nonempty sub-differential set at $\mathbf{w}$, f is called sub-differentiable at $\mathbf{w}$.

Therefore convex functions are always sub-differentiable in their domain. Next theorem shows that a local minimum of convex functions is also global minimum.

Lemma 2.2. *(Bertsekas [11])*
Let $\mathcal{C}$ be an open convex set and $\mathcal{O} \subset \mathcal{C}$ be an open set in $\mathcal{C}$. If a function $f : \mathcal{C} \mapsto \mathbb{R}$ is convex, its local minimum $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w} \in \mathcal{O}} f(\mathbf{w})$ is also a global minimum in $\mathcal{C}$.

Definition 2.4. *Strict Convex Function*
Let $\mathcal{C}$ be a convex set. A function $f : \mathcal{C} \mapsto \mathbb{R}$ is strict convex, if for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{C}$, and for any $\alpha \in [0,1]$, the following inequality holds:

$$f(\alpha \mathbf{w}_1 + (1-\alpha)\mathbf{w}_2) < \alpha f(\mathbf{w}_1) + (1-\alpha) f(\mathbf{w}_2) \quad \textit{for} \quad \mathbf{w}_1 \neq \mathbf{w}_2 \ .$$

For the strict convex functions there is at most one global minimum.

Definition 2.5. *λ-strongly Convex Function*
Let $\mathcal{C}$ be a convex set. A function $f : \mathcal{C} \mapsto \mathbb{R}$ is λ-strongly convex, if for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{C}$, and for any $\alpha \in [0, 1]$, the following inequality holds:

$$f(\alpha\mathbf{w}_1 + (1-\alpha)\mathbf{w}_2) \leq \alpha f(\mathbf{w}_1) + (1-\alpha)f(\mathbf{w}_2) - \frac{\lambda}{2}\alpha(1-\alpha)\left\|\mathbf{w}_1 - \mathbf{w}_2\right\|^2 \quad .$$

For instance, for $f(\mathbf{w}) = \lambda\left\|\mathbf{w}\right\|^2$, the following equality holds:

$$f(\alpha\mathbf{w}_1 + (1-\alpha)\mathbf{w}_2) = \alpha f(\mathbf{w}_1) + (1-\alpha)f(\mathbf{w}_2) - \lambda\alpha(1-\alpha)\left\|\mathbf{w}_1 - \mathbf{w}_2\right\|^2 \quad .$$

Therefore $f(\mathbf{w}) = \lambda\left\|\mathbf{w}\right\|^2$ is 2λ-strongly convex.

Next we define Lipschitz functions. In the book of nonlinear programming in (Bertsekas [11]), Lipschitz property is alway assumed. Without this property it is difficult to prove the convergence to a local minimum.

Definition 2.6. *ρ-Lipschitz Function*
Let $\mathcal{C} \subset \mathbb{R}^d$. A function $f : \mathbb{R}^d \mapsto \mathbb{R}$ is ρ-Lipschitz over $\mathcal{C}$, if for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{C}$, the following inequality holds:

$$\left\|f(\mathbf{w}_1) - f(\mathbf{w}_2)\right\| \leq \rho\left\|\mathbf{w}_1 - \mathbf{w}_2\right\| \quad .$$

We assume that a set of hypotheses $\mathcal{H}$ are parametrized by $\mathbf{w}$. In other words, a hypotheses in $\mathcal{H}$ is specified by each $\mathbf{w}$. Regularized Loss Minimization (RLM) rule is a learning rule which minimizes the empirical risk and a regularization function $\Omega(.)$:

$$\min_{\mathbf{w}} \left[L_{\mathcal{S}}(\mathbf{w}) + \Omega(\mathbf{w})\right] \tag{2.3}$$

$\Omega(\mathbf{w})$ is chosen so that $L_{\mathcal{S}}(\mathbf{w}) + \Omega(\mathbf{w})$ is strongly convex with respect to $\mathbf{w}$. For example, $\Omega(\mathbf{w}) = \frac{\lambda}{2}\left\|\mathbf{w}\right\|^2$, with $\lambda > 0$.

Assuming the convexity and the Lipschitz property of a loss function, the following theorem derives an upper bound of the difference between the true risk and the empirical risk of the hypothesis obtained by applying the RLM rule.

Theorem 2.1. *Corollary 13.6 in [137]*
Assume that the loss function is convex and ρ-Lipschitz. Suppose that an algorithm A solves the optimization problem (2.3) with the regularization function $\Omega(\mathbf{w}) = \frac{\lambda}{2}\left\|\mathbf{w}\right\|^2$ and outputs the hypothesis:

$$\mathbf{w}_{A(S)} = \text{argmin}_{\mathbf{w}}\left(L_{\mathcal{S}}(\mathbf{w}) + \frac{\lambda}{2}\left\|\mathbf{w}\right\|^2\right).$$

Then, it follows that

$$\mathbb{E}_{\mathcal{S}\sim\mathcal{D}^n}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)}) - L_{\mathcal{S}}(\mathbf{w}_{A(S)})\right] \leq \frac{4\rho^2}{\lambda n}.$$

We rewrite the above expected risk as:

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right] = \mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{S}}(\mathbf{w}_{A(S)})\right] + \mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)}) - L_{\mathcal{S}}(\mathbf{w}_{A(S)})\right] \tag{2.4}$$

For any vector $\mathbf{w}^*$,

$$L_{\mathcal{S}}(\mathbf{w}_{A(S)}) \leq L_{\mathcal{S}}(\mathbf{w}_{A(S)}) + \frac{\lambda}{2}\left\|\mathbf{w}_{A(S)}\right\|^2 \leq L_{\mathcal{S}}(\mathbf{w}^*) + \frac{\lambda}{2}\left\|\mathbf{w}^*\right\|^2$$

Taking expectation with respect to S,

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{S}}(\mathbf{w}_{A(S)})\right] = \mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right]$$
$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{S}}(\mathbf{w}^*) + \frac{\lambda}{2}\left\|\mathbf{w}^*\right\|^2\right] = L_{\mathcal{D}}(\mathbf{w}^*) + \frac{\lambda}{2}\left\|\mathbf{w}^*\right\|^2$$

Therefore

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right] \leq L_{\mathcal{D}}(\mathbf{w}^*) + \frac{\lambda}{2}\left\|\mathbf{w}^*\right\|^2.$$

Substituting into (2.4) and using Theorem 2.1,

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right] \leq L_{\mathcal{D}}(\mathbf{w}^*) + \frac{\lambda}{2}\left\|\mathbf{w}^*\right\|^2 + \frac{4\rho^2}{\lambda n} \tag{2.5}$$

We assume that $\|\mathbf{w}\|^2 \leq B^2$. Let $\lambda = \sqrt{\frac{4\rho^2}{B^2 n}}$. Then,

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right] \leq \min_{\mathbf{w}} L_{\mathcal{D}}(\mathbf{w}) + \frac{3\rho B}{\sqrt{n}}.$$

We have arrived the following theorem.

Theorem 2.2. *Corollary 13.9 in [137]*
Let the loss function be ρ-Lipschitz and $\mathbf{w}$ is bounded and $\|\mathbf{w}\|^2 \leq B^2$. For any training set size n, let $\lambda = \sqrt{\frac{4\rho^2}{B^2 n}}$. Then the RLM rule with the regularization function $\frac{\lambda}{2}\|\mathbf{w}\|^2$ satisfies

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right] \leq \min_{\mathbf{w}} L_{\mathcal{D}}(\mathbf{w}) + \frac{3\rho B}{\sqrt{n}}. \tag{2.6}$$

In particular, for every $\epsilon > 0$, if $n \geq \frac{9\rho^2 B^2}{\epsilon^2}$ then for every distribution $\mathcal{D}$, $\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right] \leq \min_{\mathbf{w}} L_{\mathcal{D}}(\mathbf{w}) + \epsilon$.

Based on the assumption that the loss function is convex and Lipschitz with respect to $\mathbf{w}$ and $\|\mathbf{w}\|^2 \leq B^2$, the theorem proves that the RLM rule does not over-fit. In other words, the risk of the RLM rule is always close to the minimum risk. The theorem also tells us that given a training set with size n, how the solution of RLM is close to the optimal solution in $\mathcal{H}$. The evaluation of a risk (a generalization error) based on a finite sample is a major achievement of machine learning (statistical learning theory).

2.3 Support Vector Machines

Support vector machine is one form of the RLM (Regularized Loss Minimization). In this section we derive the formulation of SVM from a geometric point of view and show how the maximal margin principle of SVM works. We also review the basic properties of kernels. In the last two sections we outline two major implementation algorithms of SVM, sequential minimization optimization (SMO) and stochastic gradient descent (SGD). The SGD also serves as an another learning rule.

2.3.1 Support Vector Machines

SVMs classify data using hyperplanes which have the maximal margin (distance) to the nearest data points. Suppose that a training set $\mathcal{S}$,

$$\mathcal{S} = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n) \ ,$$

is correctly classified by a hyperplane H_0:

$$H_0 : \quad \langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \qquad (\langle \mathbf{w}, \mathbf{x} \rangle \text{ is an inner product between } \mathbf{w} \text{ and } \mathbf{x}.),$$

that is, there exist coefficients $(\mathbf{w}, b)$ such that

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x}_i \rangle + b > 0 \qquad & \text{for } y_i = 1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b < 0 \qquad & \text{for } y_i = -1. \end{aligned}$$

Among those hyperplanes we consider two hyperplanes which are parallel to H_0:

$$\begin{aligned} H_1 \ : \quad & \langle \mathbf{w}, \mathbf{x} \rangle + b = 1 \\ H_2 \ : \quad & \langle \mathbf{w}, \mathbf{x} \rangle + b = -1. \end{aligned}$$

Since the distance between H_1 and H_2 is $\frac{2}{\|\mathbf{w}\|}$, we can obtain the optimal hyperplane with the maximal margin by solving the following quadratic optimization problem:

$$\begin{aligned} &\text{Minimize: } \frac{1}{2} \|\mathbf{w}\|^2 \\ &\text{Subject to: } \langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 \qquad \text{for } y_i = 1 \quad i = 1, \ldots, n \\ &\qquad\qquad \langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq -1 \qquad \text{for } y_i = -1 \quad i = 1, \ldots, n \end{aligned}$$

The last two constraints can be combined into one set of inequalities:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \qquad \text{for} \quad i = 1, \ldots, n$$

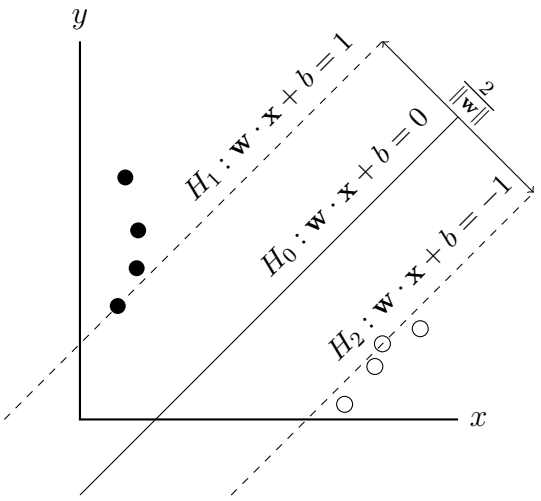


Figure 2.5: Margin between H_1 and H_2

(Burges [20]).

This maximal margin principle is the most important idea in SVMs. Based on this model two modifications are added, which introduce hyper-parameters to SVMs. In order to allow some misclassification in the data with some noise, slack variables ξ_i are introduced to relax the constraint $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ to $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$, $\xi_i \geq 0$. Nonlinear feature maps Φ are also introduced to map the input data $\mathbf{x} \in X$ into a Hilbert space H.

$$\begin{aligned} &\text{Minimize: } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \\ &\text{Subject to: } y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \\ &\qquad \xi_i \geq 0 \qquad\qquad \text{for } i = 1, \ldots, n \end{aligned} \tag{2.7}$$

The maximal margin principle of SVM has shown the excellent prediction accuracy for the real-world problems. More importantly the generalization error rate of SVM does not depend on the dimension of the spaces, which is stated in the following theorem.

Theorem 2.3. *(Shalev-Shwartz and Ben-David [137])*
Suppose that $\|\Phi(\mathbf{x})\|_2 \leq R$, $\|\mathbf{w}\|_2 \leq B$, *and for any* y, *the loss function* $\ell(\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle, y)$ *is* ρ*-Lipschitz, and* $|\ell(\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle, y)| < c$ *with respect to any values of* $\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle$. *Then, for any* $\delta \in (0, 1)$, *with probability of at least* $1 - \delta$

$$L_{\mathcal{D}}(\mathbf{w}) \leq L_{\mathcal{S}}(\mathbf{w}) + \frac{2\rho BR}{\sqrt{n}} + c\sqrt{\frac{2 \ln\left(\frac{2}{\delta}\right)}{n}}$$

over an i.i.d. sample of size n.

The product BR almost always appears in the theorems for the error rate of SVM.

The optimization problem in (2.7) is one form of RLM. In general, the standard SVM can be formulated as follows:

$$\min_{\mathbf{w}} \left(\Omega(\mathbf{w}) + C \sum_{i=1}^{n} \ell\left(\mathbf{w}, \Phi(\mathbf{x}_i), y_i\right) \right) \tag{2.8}$$

where $\Omega(\mathbf{w})$ is a regularization function; and ℓ is a loss function for the hypothesis based on the parameter $\mathbf{w}$.

A common choice of the regularization function is the ℓ_2-norm $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2$ which makes the optimization problems strongly convex. $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|$ is also frequently used because it induces sparse solutions, which is called ℓ_1-norm.

In binary classification problems, $y_i \in \{-1, 1\}$ and the loss function in (2.7) is the hinge loss function, which is also written as

$$\ell\left(\mathbf{w}, \Phi(\mathbf{x}_i), y_i\right) = \max(0, 1 - y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b)) \ . \tag{2.9}$$

In multi-class classification problems, where $y_i \in \{1, \ldots, l\}$, we prepare a weight vector for each class, so that

$$\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_l] \, .$$

A common choice of the loss function for multi-class classification is the generalized hinge loss function, which is defined as follows:

$$\ell\left(\mathbf{w}, \Phi(\mathbf{x}_i), y_i\right) = 1 + \max_{h \neq y_i} \langle \mathbf{w}_h, \Phi(\mathbf{x}_i)\rangle - \langle \mathbf{w}_{y_i}, \Phi(\mathbf{x}_i)\rangle \ . \tag{2.10}$$

The optimization problem can be equivalently formulated as follows (Crammer and Singer [30]):

$$\begin{aligned} &\min_{\mathbf{w}, \boldsymbol{\xi}} \frac{1}{2} \sum_{h=1}^{l} \|\mathbf{w}_h\|^2 + C \sum_{i=1}^{n} \xi_i \\ &\text{s.t.} \ \ \langle \mathbf{w}_{y_i}, \Phi(\mathbf{x}_i)\rangle - \max_{h \neq y_i} \langle \mathbf{w}_h, \Phi(\mathbf{x}_i)\rangle \geq 1 - \xi_i \\ &\qquad \text{for } i = 1, \ldots, n; \ \ h = 1, \ldots, l \end{aligned} \tag{2.11}$$

The optimization problem (2.11) means that if $\|\mathbf{w}_{y_i}\| = 1$, then $\langle \mathbf{w}_{y_i}, \Phi(\mathbf{x}_i)\rangle$ is the length of the projection of $\Phi(\mathbf{x}_i)$ onto $\mathbf{w}_{y_i}$. The above inequalities require that the direction of $\mathbf{w}_{y_i}$ is closer to that of $\Phi(\mathbf{x}_i)$ than any other $\mathbf{w}_{h \neq y_i}$. Instead of fixing $\|\mathbf{w}_h\| = 1, h = 1, \ldots, l$ and maximizing the difference between $\langle \mathbf{w}_{y_i}, \Phi(\mathbf{x}_i)\rangle$ and $\max_{h \neq y_i} \langle \mathbf{w}_h, \Phi(\mathbf{x}_i)\rangle$,

(2.11) minimizes $\sum_{h=1}^{l} \|\mathbf{w}_h\|^2$ ($\|\mathbf{w}_h\|^2, h = 1, \ldots, l$) while fixing the minimum difference between $\langle \mathbf{w}_{y_i}, \Phi(\mathbf{x}_i) \rangle$ and $\max_{h \neq y_i} \langle \mathbf{w}_h, \Phi(\mathbf{x}_i) \rangle$ to be 1.

Support Vector Data Descriptions (SVDDs) were introduced in (Tax and Duin [154]) and revised in (Chang et al. [165]) as one-class support vector machines for detecting outliers:

$$\begin{aligned} \min_{R, \mathbf{a}, \boldsymbol{\xi}} \quad & R + C \sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & \|\Phi(\mathbf{x_i}) - \mathbf{a}\|^2 \leq R + \xi_i \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \ . \end{aligned} \tag{2.12}$$

where $\mathbf{a}$ and R are the center and the radius of the sphere that contains all the $\Phi(\mathbf{x_i})$, $i = 1, \ldots, n$. In this case, the loss function is written as:

$$\ell\left(\mathbf{a}, R, \Phi(\mathbf{x}_i), y_i\right) = \max(0, \|\Phi(\mathbf{x_i}) - \mathbf{a}\|^2 - R) \ .$$

2.3.2 Duality for Convex Functions

In this section, we review a duality form for convex functions. Let us consider the following form of the optimization problem:

$$\begin{aligned} P: \quad \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \end{aligned} \tag{2.13}$$

where $f(\mathbf{x})$ is called the objective function and $g_i(\mathbf{x})$, $i = 1, \ldots, m$, are called the constraint functions.

Slater's constraint qualification: For the problem (2.13), if there exists a $\mathbf{x}^*$ such that $\mathbf{x}^*$ satisfies the following inequality:

$$g_i(\mathbf{x}^*) < 0, \quad i = 1, \ldots, m, \tag{2.14}$$

Slater's constraint qualification is satisfied.

For convex functions, the Wolfe dual is commonly used. SVMs are originally solved in the dual form.

Wolfe dual: For the optimization problem P in (2.13), the Wolfe dual is defined as:

$$\begin{aligned} WD: \quad & \max_{\mathbf{u}} \quad f(\mathbf{u}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{u}) \\ & \text{s.t.} \quad \nabla f(\mathbf{u}) + \nabla \sum_{i=1}^{m} \lambda_i g_i(\mathbf{u}) = 0 \\ & \qquad \lambda_i \geq 0, \quad i = 1, \ldots, m. \end{aligned} \tag{2.15}$$

Lemma 2.3. *(Bector et al. [9], Mond [110])*
If $f + \sum_{i=1}^{m} \lambda_i g_i$, for $\lambda_i \geq 0$, is convex, the weak duality ($\min_{\mathbf{x}} [f(\mathbf{x})] \geq \max_{\mathbf{u}} [f(\mathbf{u}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{u})]$) holds. If the optimization problem P has an optimal solution $\mathbf{x} = \mathbf{x}_0$ and a constraint qualification such as Slater's condition (2.14) is satisfied, then there exists a $\boldsymbol{\lambda}_0$, such that $(\mathbf{u} = \mathbf{x}_0, \boldsymbol{\lambda}_0)$ is optimal for WD, and strong duality ($\min_{\mathbf{x}} [f(\mathbf{x})] = \max_{\mathbf{u}} [f(\mathbf{u}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{u})]$) holds.

2.3.3 Kernels

The Wolfe dual of the optimization problem (2.7) is written as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \ . \end{aligned} \tag{2.16}$$

In the dual form, the mapping $\Phi(\mathbf{x})$ appears only in the inner products $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$, which we call a *kernel*. Let K be a matrix whose elements are inner products of $\Phi(\mathbf{x})$. That is,

$$K(i, j) = \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle, \quad i, j = 1, \ldots, n,$$

which we call a *kernel matrix*.

A kernel can be constructed by the inner product of functions. On the other hand, if a function satisfies the following Positive Definite Symmetric condition, it can be expressed as the inner product of some functions, and is therefore a kernel.

Definition 2.7. *Positive Definite Symmetric (PDS) Functions (Cortes et al. [27])*
A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is PDS on a set $\mathcal{X}$ if it is symmetric:

$$k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x}) \ ,$$

for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, and

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

for all $n \geq 0$, $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathcal{X}$, *and* $\{a_1, \ldots, a_n\} \in \mathbb{R}$.

Theorem 2.4. *Mercer's Condition (Steinwart and Christmann [148])*
A function $k : X \times X \to \mathbb{R}$ *is a kernel if and only if it is PDS.*

In this thesis, we will use the following kernels.

- Gaussian kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right) \tag{2.17}$$

- Polynomial kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = (a\langle \mathbf{x}_i, \mathbf{x}_j \rangle + b)^c, a > 0, \, b \geq 0, \, c = 1, 2, \ldots \tag{2.18}$$

If a kernel $k(\mathbf{x}, \mathbf{y})$ is expressed as $k(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x} - \mathbf{y})$ with some function κ. it is called a shift-invariant kernel. Gaussian kernel is a shift-invariant kernel.

New kernels can be constructed based on other kernels. For instance, given any two kernels k_1 and k_2 on a set $\mathcal{X}$, ak_1 $(a \geq 0)$ and $k_1 + k_2$ are also kernels on $\mathcal{X}$.

Theorem 2.5. *Products of Kernels (Steinwart and Christmann [148])*
Let k_1 *be a kernel on* $\mathcal{X}_1$ *and* k_2 *be a kernel on* $\mathcal{X}_2$. *Then, the product* $k_1 k_2$ *is a kernel on* $\mathcal{X}_1 \times \mathcal{X}_2$.

The product of two kernel matrices is not necessarily a kernel matrix (Meenakshi and Rajian [104]). However, from Theorem 2.5, the Hadamard product (point-wise product) of two kernel matrices is also a kernel matrix in the corresponding tensor product space. It is also shown that $\exp(k(\mathbf{x}, \mathbf{y}))$ is a kernel if $k(\mathbf{x}, \mathbf{y})$ is a kernel (Cortes et al. [27]).

The dual form is not the only method to solve the optimization problem (2.8). Let's consider the following general problem:

$$\min_{\mathbf{w}} \left(\langle \mathbf{w}, \Phi(\mathbf{x}_1) \rangle, \ldots, \langle \mathbf{w}, \Phi(\mathbf{x}_1) \rangle\right) + g\left(\|\mathbf{w}\|\right) \tag{2.19}$$

where $g : \mathbb{R}_+ \mapsto \mathbb{R}$ is a monotonically nondecreasing function. The problem (2.8) is a realization of (2.19).

Theorem 2.6. *Representer Theorem (Shalev-Shwartz and Ben-David [137])*
Let $\mathbf{w}^*$ *be an optimal solution of (2.19). Then there exists a vector* $\alpha \in \mathbb{R}^n$ *such that* $\mathbf{w}^* = \sum_{i=1}^n \alpha_i \Phi(\mathbf{x}_i)$.

Using the representer theorem, SVMs, presented in (2.8), can be solved more efficiently using "Stochastic Gradient Descent" (SGD) method which we describe in Section 2.3.5.

2.3.4 Sequential Minimization Optimization (SMO)

In this section, we outline the Sequential Minimization Optimization (SMO) which is a standard method to solve the Wolfe dual of SVMs (2.16).

SMO makes use of the equality constraint:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.20}$$

to efficiently solve (2.16). For $i, j \in \{1, \ldots, n\}, i \neq j$, SMO solves the optimal (α_i^*, α_j^*) while fixing other $\alpha_k, k \neq i, j$. Assume that $i = 1$ and $j = 2$. When $y_1 \neq y_2$, (2.20) reduces to $\alpha_2 = \alpha_1 + k$ where k is a constant. $\alpha_1 + k$ is a increasing function of α_1 and α_2 takes the minimum $k = \alpha_2 - \alpha_1$ at $\alpha_1 = 0$. Under the condition that $0 \leq \alpha_i \leq C$, the lower bound L of α_2 is given as $L = \max(0, \alpha_2 - \alpha_1)$. Similarly, the upper bound H of α_2 is given as $H = \min(C, C + \alpha_2 - \alpha_1)$. When $y_1 = y_2$, $L = \max(0, \alpha_2 + \alpha_1 - C)$ and $H = \min(C, \alpha_2 + \alpha_1)$.

α_2^{new} is analytically solved as

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_2 - E_1)}{K(1,1) + K(2,2) - 2K(1,2)} \tag{2.21}$$

where $E_i = \sum_{j=1}^{n} \alpha_j y_j K(j, i)$ (Platt [120]). If α_2^{new} is outside of the bound $[L, H]$, we project it into the range $[L, H]$.

$$\alpha_2^{new,clipped} = \begin{cases} H & \text{if } \alpha_2^{new} > H \\ \alpha_2^{new} & \text{if if } L \leq \alpha_2^{new} \leq H \\ L & \text{if } \alpha_2^{new} < L \end{cases}$$

The value of α_1^{new} is computed as

$$\alpha_1^{new} = \alpha_1 + y_1 y_2 (\alpha_2 - \alpha_2^{new,clipped}).$$

2.3.5 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) not only provides a means to solve the standard formulation of SVMs (2.8), but also it serves as an another learning paradigm. Let's start with the review of gradient. For a differentiable function $f : \mathbb{R}^d \mapsto \mathbb{R}$ at $\mathbf{w}$, let $\nabla f(\mathbf{w})$ denote the gradient vector of the function, which is the vector of partial derivatives, i.e.

$$\nabla f(\mathbf{w}) = \left(\frac{\partial f(\mathbf{w})}{\partial w_1}, \ldots, \frac{\partial f(\mathbf{w})}{\partial w_d} \right) \quad . \tag{2.22}$$

SVMs (2.8) with the hinge loss or the generalized hinge loss are not differentiable but they are sub-differentiable. The sub-gradients of SVMs (2.8) can be computed using the following lemmas.

Lemma 2.4. *If f is differentiable at $\mathbf{w}$, $\partial f(\mathbf{w})$ contains only one element and $\partial f(\mathbf{w}) = \{\nabla f(\mathbf{w})\}$*

Lemma 2.5. *Sub-gradient of Pointwise Maximum Functions (Shalev-Shwartz and Ben-David [137])*
Let $f(\mathbf{w}) = \max_i f_i(\mathbf{w})$ for differentiable convex functions at $f_i(\mathbf{w})$ for $i = 1, \ldots, l$. Given a $\mathbf{w}$, let $j \in \operatorname{argmax}_i f_i(\mathbf{w})$. Then

$$\nabla f_j(\mathbf{w}) \in \partial f(\mathbf{w}).$$

In the standard gradient descent, in order to minimize a differentiable convex function $f(\mathbf{w})$, we start with an initial value of $\mathbf{w}^1$ and take a step in the direction of the negative gradient:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla f(\mathbf{w}).$$

In SGD, the descent direction is a random vector and its expectation is a sub-gradient of the function $f(\mathbf{w})$ at the current $\mathbf{w}^t$:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \mathbf{v}_t \tag{2.23}$$

$$\mathbb{E}\left[\mathbf{v}_t \mid \mathbf{w}^t\right] \in \partial f(\mathbf{w}^t) \tag{2.24}$$

After T iteration, we estimate the optimal $\mathbf{w}$ as

$$\bar{\mathbf{w}} = \frac{1}{T} \sum_{i=1}^{T} \mathbf{w}^t. \tag{2.25}$$

The following theorem shows that if we choose the descent direction to satisfy the condition (2.24), then $\bar{\mathbf{w}}$ approach the optimal solutions. It also provides a means to automatically compute the step size η for the SGD algorithm.

Theorem 2.7. *Theorem 14.8 in [137]*
Let $B, \rho > 0$. Let f be a convex function and let

$$\mathbf{w}^* \in \operatorname{argmin}_{\mathbf{w}:\|\mathbf{w}\|<B} f(\mathbf{w}).$$

Assume that SGD is run for T iterations with $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$. Assume also that for all t, $\|\mathbf{v}_t\| \leq \rho$ with probability 1. Then,

$$\mathbb{E}\left[f(\bar{\mathbf{w}})\right] - f(\mathbf{w}^*) \leq \frac{B\rho}{\sqrt{T}}.$$

Therefore, for any $\epsilon > 0$, to achieve $\mathbb{E}[f(\bar{\mathbf{w}})] - f(\mathbf{w}^*) \leq \epsilon$, it suffices to run the SGD algorithm for a number of iterations that satisfies

$$T \geq \frac{B^2\rho^2}{\epsilon^2}$$

Using SGD we can directly minimize the risk function $L_{\mathcal{D}}(\mathbf{w})$:

$$L_{\mathcal{D}}(\mathbf{w}) := \mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left[\ell\left(\mathbf{w}, \mathbf{z}\right)\right] \quad .$$

The random direction $\mathbf{v}_t$ is obtained as a sub-gradient of $\ell(\mathbf{w}, \mathbf{z})$ at $\mathbf{w}^t$ with a fresh sample $\mathbf{z}$.

To see this, let $\ell(\mathbf{w}, \mathbf{z})$ be a convex loss function with a sub-gradient $\mathbf{v}_t \in \partial f(\mathbf{w})$. Then,

$$\ell(\mathbf{u}, \mathbf{z}) - \ell\left(\mathbf{w}^t, \mathbf{z}\right) \geq \langle \mathbf{u} - \mathbf{w}^t, \mathbf{v}_t \rangle \tag{2.26}$$

Using this sub-gradient $\mathbf{v}_t$,

$$\begin{aligned} L_{\mathcal{D}}(\mathbf{u}) - L_{\mathcal{D}}(\mathbf{w}^t) &= \mathbb{E}\left[\ell(\mathbf{u}, \mathbf{z}) - \ell\left(\mathbf{w}^t, \mathbf{z}\right) \mid \mathbf{w}^t\right] \\ &\geq \mathbb{E}\left[\langle \mathbf{u} - \mathbf{w}^t, \mathbf{v}_t \rangle \mid \mathbf{w}^t\right] \\ &= \langle \mathbf{u} - \mathbf{w}^t, \mathbb{E}\left[\mathbf{v}_t \mid \mathbf{w}^t\right] \rangle \end{aligned} \tag{2.27}$$

Therefore, the expectation of $\mathbf{v}_t$ ($\mathbb{E}[\mathbf{v}_t \mid \mathbf{w}^t]$) is a sub-gradient of $L_{\mathcal{D}}(\mathbf{w})$ at $\mathbf{w}^t$.

When we apply SGD to SVM with the training data $\mathcal{S}$ with the size n, we can evaluate the estimate of SGD in two steps:

1. Using Theorem 2.2, given a training size n we can compute how the solution of SVM (2.8) is close to the optimal solution with the assumption $\|\mathbf{w}\|^2 < B^2$.

2. Using Theorem 2.7, given a training set $\mathcal{S}$ and the iteration size T, we can compute how the solution of SGD is close to the solution of SVM (2.8).

An advantage of SGD over SMO is that the program code for SGD is much simpler than SMO and more suitable for solving a large scale data. An advantage of SMO over SGD is that the SMO is more accurate for a problem with a small data because SMO solves (2.8) directly whereas SGD estimates the solution via the stochastic sampling.

2.4 Evolutionary Computation

In this section we review some basic concepts and algorithms in Evolutionary computation (EC). EC is a family of population-based algorithms for global optimization, which is a subfield of artificial intelligence. It covers two main classes of nature-inspired algorithms, evolutionary algorithms (EAs) and swarm intelligence (SI).

Evolutionary algorithms (EA)

Evolutionary algorithms (EA) are generic population-based algorithms which were developed in different places since 1960s (Vikhar [164]). In 1960s, Rechenberg introduced "evolutionary strategies" to optimize real-valued parameters for devices such as air-foils. Fogel, Owen, and Walsh introduced "evolutionary programming in which candidate solutions are represented as a finite-state machines (Mitchell [109]). EA also includes most popular algorithms; Genetic Algorithms (GA) and Genetic Programming (GP).

Genetic Algorithms (GA) Genetic Algorithms (GA) were invented by John Holland and developed in 1960s and 1970s at the University of Michigan by Holland and his students and colleagues (Holland [64], [65]). In the GA, individuals which are also called "chromosomes" are represented by bit strings and evolved through the mechanisms inspired by biological evolution; selection, reproduction, crossover and mutation. This set of evaluational operations is a major innovation introduced by GA (Mitchell [109]). The applications of GA in Machine learning include evolutions of weights for neural networks, rules for learning classifier systems, and sensors for robots (Mitchell [109]).

Genetic Programming (GP) Genetic Programming (GP) was invented by John Koza and established by four books by Koza starting in 1992 (Koza [81]). In GP, individuals are computer programs and represented by the parse trees of the program codes. The computer programs are evolved through the mechanisms introduced by GA. Koza [81] shows that a number of real-world problems in many different fields can be expressed by a set of computer programs and solved by searching for the optimal solutions. Koza [83] lists 77 results where Genetic Programming is human competitive. GP also provides a powerful and flexible means to solve classification problems (Espejo et al. [43]).

Swarm Intelligence (SI)

Swarm Intelligence (SI) is inspired by the collective intelligence of social animals. The SI systems consist of a population of simple agents interacting with each other and with their environment. The agents follow simple rules and interactions between agents lead to the emergence of "intelligent" global behavior without no centralized

control (Bonabeau et al. [13]). Two main algorithms in SI are Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). ACO is an algorithm inspired by the behavior of ant colony. The transition probability of an ant from a town i to a town j is computed as a function of distance between the two towns and the "pheromone" quantity remaining on the edge (i, j) (Dorigo et al. [38]). ACO is suitable to the problems that need to find the shortest path to the goals (Dorigo and Gambardella [37]).

Particle Swarm Optimization (PSO) Particle Swarm Optimization (PSO) was invented by James Kennedy and Russell Eberhart (Kennedy and Eberhart, [70]). It consists of candidate solutions, called particles. In PSO a population of particles is called a swarm. Particles are moving around the search space while exchanging the information about the best position in its own history and the best position in its neighbor's history. PSO is more suitable to the problems which compute the optimal solutions in continuous spaces, compared to GA and GP. It does not assume differentiability and provides a means to find optimal solutions when the standard ML methods can not be applied.

2.5 Particle Swarm Optimization (PSO)

In this section, we start with the introduction of PSO. Then we review some common models of continuous PSO, binary PSO and discrete PSO, which will be used in the experiments in the following chapters.

2.5.1 Introduction

In PSO, each particle keeps the information of its position and its velocity. The position of a particle is a point in a space, usually Euclidean space $\mathbb{R}^{d'}$. The velocity is a vector which determines the position of the particle in the next iteration. Each particle also keeps the best position of its own history, which is called "pbest" and the best position of the swarm's history (its neighbor's history), which is called "gbest".

Let $\mathbf{x}^t$ and $\mathbf{v}^t$ be the position and the velocity of a particle at iteration t. Then $\mathbf{v}^{t+1}$ is determined as a linear combination of $\mathbf{v}^t$, $(pbest - \mathbf{x}^t)$ and $(gbest - \mathbf{x}^t)$ (Figure 2.6).

Algorithm 2.1 describes the main steps of PSO. In the following sections we will see the position update rule and the velocity update rule for each model.

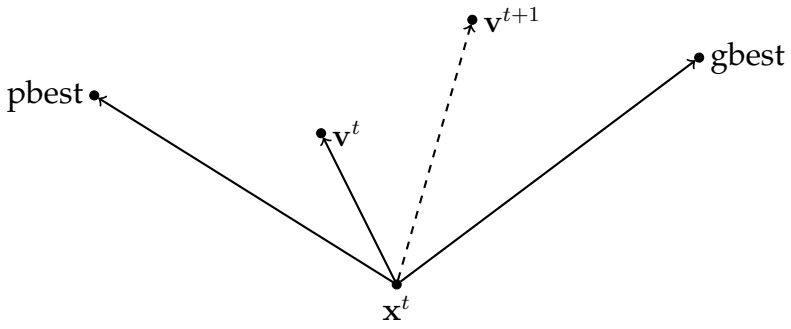


Figure 2.6: Velocity

Algorithm 2.1: Main Steps of PSO

Initialization:
Initialize positions and velocities of particles
Evaluate positions of particles
Compute $pbest$
Compute $gbest$
while (termination conditions are not satisfied) **do**
  Update velocities of particles
  Update positions of particles
  Evaluate positions of particles
  Update $pbest$
  Update $gbest$
end while

2.5.2 Continuous PSO

In general, we consider the following optimization problem for continuous PSO:

$$\begin{aligned} &\text{minimize:} F(\mathbf{x}) \\ &\mathbf{x} = (x_1, \ldots, x_{d'}) \in S \subseteq \mathbb{R}^{d'} \end{aligned} \tag{2.28}$$

where the search space S is a hyper-cube in $\mathbb{R}^{d'}$,

$$l_j \leq x_j \leq u_j, \qquad l_j, u_j \in \mathbb{R}, \quad j = 1, \ldots, d'. \tag{2.29}$$

(Note that we use the number of dimension d' for PSO to distinguish the number of features d in the input space in SVMs in Section 2.3.1.)

In the following sections we review two standard models of continuous PSO.

Standard PSO (SPSO)

Kennedy and Eberhart [70] firstly invented the original form of PSO. As depicted in Figure 2.6, the velocity on the next iteration is determined by the linear combination of the current velocity, the vector toward pbest and the vector toward gbest. At iteration t, the j-th dimension of position and velocity for the particle i are updated according to the following formulas:

$$v_{i,j}^{t+1} = v_{i,j}^t + c_1 r_{1,i,j}^t \left(p_{i,j}^t - x_{i,j}^t\right) + c_2 r_{2,i,j}^t \left(g_j^t - x_{i,j}^t\right) \tag{2.30}$$

$$x_{i,j}^{t+1} = x_{i,j}^t + v_{i,j}^{t+1}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, d' \tag{2.31}$$

$r_{1,i,j}^t$ and $r_{2,i,j}^t$ are random numbers distributed uniformly in $[0, 1]$ for each i, j and t. c_1 and c_2 are constant values. The pbest for a particle i and the gbest for the minimization problem (2.28) are updated as follows.

$$\mathbf{p}_i^{t+1} = \begin{cases} \mathbf{x}_i^{t+1} & \text{if } F(\mathbf{x}_i^{t+1}) < F(\mathbf{p}_i^t) \\ \mathbf{p}_i^t & \text{otherwise} \end{cases} \tag{2.32}$$

$$\mathbf{g}^{t+1} = \text{argmin}_{\mathbf{p}_i^{t+1}} F(\mathbf{p}_i^{t+1}) \tag{2.33}$$

Shi and Eberhart [140] introduced the inertia weight ω in the velocity update formula, which controls the effect of the velocity of the previous iteration. It is the most popular form of PSO, which we call the standard PSO (SPSO) in this article.

$$v_{i,j}^{t+1} = \omega v_{i,j}^t + c_1 r_{1,i,j}^t \left(p_{i,j}^t - x_{i,j}^t\right) + c_2 r_{2,i,j}^t \left(g_j^t - x_{i,j}^t\right) \tag{2.34}$$

Fully Informed PSO (FIPS)

Mendes et al. [106] proposed the fully informed particle swarm optimizer FIPS, in which each particle can share information with its neighbors using various topologies rather than only the "gbest" topology in SPSO. The velocity update rule of FIPS is formulated as:

$$\mathbf{v}_i^{t+1} = \omega \mathbf{v}_i^t + \sum_{k \in T_i^t} \frac{c_k}{|T_i^t|} \langle \mathbf{r}_{k,i}^t, \mathbf{p}_k^t - \mathbf{x}_i^t \rangle \tag{2.35}$$

where T_i^t is the set of indices of the neighbors of the particle i and $|T_i^t|$ is the number of the particles in T_i^t. $\mathbf{r}_{k,i}^t$ is a vector of random numbers uniformly generated in $[0,1]^{d'}$.

2.5.3 Binary PSO

In this section we review four binary PSO models.

Original Binary PSO (OBPSO)

The extension of PSO for discrete problems was firstly introduced by Kennedy and Eberhart [72], which we call the original binary PSO (OBPSO). In binary PSO, the positions of particles are restricted to the vertices of hyper-cubes, which are represented by binary values 0 or 1. The velocity $v_{i,j}^t$ is interpreted as the probability of $x_{i,j}^t$ taking the value 1.

$$v_{i,j}^{t+1} = v_{i,j}^t + c_1 r_{1,i,j}^t \left(p_{i,j}^t - x_{i,j}^t\right) + c_2 r_{2,i,j}^t \left(g_j^t - x_{i,j}^t\right) \tag{2.36}$$

which is the same form as in (2.30). Since the velocity represents a probability, it should be constrained to the interval $[0,1]$. Using a sigmoid function $\Lambda(v) = \frac{1}{1+e^{-v}}$, the position update rule is defined as

$$x_{i,j}^{t+1} = \begin{cases} 1 & \text{if } r < \Lambda(v_{i,j}^{t+1}) \\ 0 & \text{otherwise} \end{cases} \tag{2.37}$$

where r is a random number generated uniformly in $[0,1]$.

V-shaped Binary PSO (VBPSO)

Since the invention of the original binary PSO, many variants of binary PSO were proposed. Mirjalili and Lewis [108] proposed "V-shaped functions" such as $V(v) =$

$\frac{2}{\pi}\arctan(\frac{2}{\pi}v)$ for the transformation function in (2.37). For the V-shaped functions, the position update rule is defined as

$$x_{i,j}^{t+1} = \begin{cases} \bar{x}_{i,j}^{t} & \text{if } r < V(v_{i,j}^{t+1}) \\ x_{i,j}^{t} & \text{otherwise} \end{cases} \tag{2.38}$$

where $\bar{x}$ is a complement operator, such that, $\bar{x}_{i,j}^{t} = 0$, if $x_{i,j}^{t} = 1$, and $\bar{x}_{i,j}^{t} = 1$, if $x_{i,j}^{t} = 0$. We denote this model as VBPSO.

Khanesar's Binary PSO (KBPSO)

Khanesar et al. [75] proposed an another variant of binary PSO, in which the velocity of a particle is the probability that the particle changes the position. They introduced two vectors $\mathbf{v}_{0,i}^{t}$ and $\mathbf{v}_{1,i}^{t}$, which can be interpreted as a probability of the bits of particles changing to 0 and 1 respectively, conditional on the current values of the bits in the particle. The velocity is then defined as

$$v_{i,j}^{t} = \begin{cases} v_{0,i,j}^{t}, & \text{if } x_{i,j}^{t} = 1 \\ v_{1,i,j}^{t}, & \text{if } x_{i,j}^{t} = 0 \ . \end{cases}$$

where

$$\begin{aligned} v_{0,i,j}^{t+1} &= \omega \ v_{0,i,j}^{t} + d_{0,1,i,j}^{t} + d_{0,2,i,j}^{t} \ ; \\ v_{1,i,j}^{t+1} &= \omega \ v_{1,i,j}^{t} + d_{1,1,i,j}^{t} + d_{1,2,i,j}^{t} \ , \end{aligned}$$

and

$$\begin{aligned} d_{0,1,i,j}^{t} &= -c_1 r_{1,i,j}^{t}, \quad d_{1,1,i,j}^{t} = c_1 r_{1,i,j}^{t} \ , \quad \text{if } p_{i,j}^{t} = 1 \\ d_{0,1,i,j}^{t} &= c_1 r_{1,i,j}^{t}, \quad d_{1,1,i,j}^{t} = -c_1 r_{1,i,j}^{t} \ , \quad \text{if } p_{i,j}^{t} = 0 \\ d_{0,2,i,j}^{t} &= -c_2 r_{2,i,j}^{t}, \quad d_{1,2,i,j}^{t} = c_2 r_{2,i,j}^{t} \ , \quad \text{if } g_{j}^{t} = 1 \\ d_{0,2,i,j}^{t} &= c_2 r_{2,i,j}^{t}, \quad d_{1,2,i,j}^{t} = -c_2 r_{2,i,j}^{t} \ , \quad \text{if } g_{j}^{t} = 0. \end{aligned}$$

The position update rule is defined as:

$$x_{i,j}^{t+1} = \begin{cases} \bar{x}_{i,j}^{t}, & \text{if } r < \Lambda(v_{i,j}^{t+1}) \\ x_{i,j}^{t}, & \text{otherwise} \end{cases}$$

where $\Lambda(v)$ is the sigmoid function $\Lambda(v) = \frac{1}{1+e^{-v}}$. In our experiments we denote this model as KBPSO.

Sticky Binary PSO (SBPSO)

The velocity update formula (2.34) in Standard PSO (SPSO) utilizes three factors— its *momentum* (the inertia term ωv), the difference from pbest and the difference from gbest to determine the next movement. The momentum corresponds to the tendency to keep the current velocity. Nguyen et al. [115] proposed the following velocity formula:

$$v_j = i_m \times w_{m,j} + i_p \times |pbest_j - x_j| + i_g \times |gbest_j - x_j|.$$

where (i_m, i_p, i_g) are set to some constants such that $i_m + i_p + i_g = 1$. The weight of the moment $w_{m,j}$ is determined so that when the bit in the dimension j is just flipped it has a small probability to change its state and the probability increases linearly over the successive iterations.

$$w_{m,j} = \min\left(1, \frac{\text{currentLife}_j}{\text{maxLife}}\right)$$

where currentLife$_j$ is the number of iteration since the bit in the dimension j has been flipped and maxLife is a positive constant. The position update rule for the particle i is defined as:

$$x_{i,j}^{t+1} = \begin{cases} \bar{x}_{i,j}^t, & \text{if } r < v_{i,j}^{t+1} \\ x_{i,j}^t, & \text{otherwise.} \end{cases}$$

In our experiments we denote this model as SBPSO (Sticky Binary PSO).

2.5.4 Discrete PSO

In this section we review one discrete PSO model.

Izakian's Discrete PSO (IDPSO)

Izakian et al. [67] proposed a discrete PSO with m responses, which extends the binary PSO model in Section 2.5.3. In their model a position of particles is represented as a $m \times d'$ matrix whose elements $x(k, j)$ have either a value of 0 or 1. and in each column only one element is 1 and other elements are 0. The velocity update formula is defined as:

$$\begin{aligned} v_i(k,j)^{t+1} =& v_i(k,j)^t + c_1 r_{1,i,j}^t \left(p_i(k,j)^t - x_i(k,j)^t\right) \\ &+ c_2 r_{2,i,j}^t \left(g(k,j)^t - x_i(k,j)^t\right) \end{aligned} \tag{2.39}$$

for each particle i. The position update formula is defined as:

$$x_i(k,j)^{t+1} = \begin{cases} 1 & \text{if } v_i(k,j)^{t+1} = \max_k v_i(k,j)^{t+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.40}$$

The equation (4.17) means that in each column of the position matrix, 1 is allocated to the element which corresponds to the maximum velocity. In our experiments we denote this model as IDPSO.

2.6 Genetic Algorithm

A system of Genetic Algorithm (GA) consists of individuals which are often represented as bit strings. The hypothesis represented by these strings can be very complex, such as the if-then rule, conjunctions and disjunctions (Grefenstette [56], De Jong [32]).

The whole set of individuals is called a population. GA optimizes the population according to a fitness function. The search space is explored using the genetic operators (Holland [65]). Algorithm 2.2 describes the main steps of GA.

Algorithm 2.2: Main Steps of GA

```
Initialization:
Creation of Initial Population
Fitness Evaluation
while (termination conditions are not satisfied) do
  Selection
  Genetic Operation (crossover, mutation, reproduction)
  Fitness Evaluation
end while
```

Creation of Initial Population

The first step of the GA is to create an initial population. The bit strings in the initial population are usually created randomly.

Fitness Evaluation and Selection Mechanism

Fitness is a measure to evaluate the goodness of each individual. It measures the performance of each individual in absolute terms or relative to other individuals. The

fitness of each individual affects the probability of its selection and survival.

The GA can be used with different selection methods. In roulette wheel selection, the probability of selecting an individual is given by the ratio of its fitness to the fitness of other members of the current population. Tournament selection is also a popular selection method. There are several options in the method:

- $k(k > 1)$ individuals are selected randomly, and the individual with the highest fitness is selected (Banzhaf et al. [3]).
- Two individuals are selected randomly and the individual with the better fitness is selected with a pre-defined probability p (Mitchell [109]).

The tournament selection often yields a more diversity in the population than the roulette wheel selection (Mitchell [109]).

As an option, we can use "Elitism". The elitism forces the GA to retain some number of best individuals at each generation. Many researches found that the elitism significantly improves the performance of GA (Mitchell [109]).

Genetic Operation

Genetic operators makes change to the bit string of individuals. They are primal ways to explore the search space and keep diversity in the population. The next population is determined by a set of operations such as crossover, mutation and reproduction over selected members in the current population.

Crossover: The crossover operation produces two new offspring from two parent strings. There are several types of crossover operations which are determined by an additional string called the crossover mask. The crossover mask is a string with 0s and 1s. The substrings corresponding to the positions of 1s are exchanged during the crossover operation.

- Single-point crossover: In the one-point crossover, we select i-th position in the two strings of n bits. We swap the substrings between $(i + 1)$-th position and n-th position. The crossover mask is a string which consists of i contiguous 0s and $(n - i)$ contiguous 1s.
- Two-point crossover: In the two-point crossover, we select i-th position and j-th position of the two strings and substrings between $(i + 1)$-th position and

j-th position are swapped. The crossover mask is a string which consists of i contiguous 0s, $(j-i)$ contiguous 1s and $(n-j)$ contiguous 0s.

- Uniform crossover: Uniform crossover combines bits sampled uniformly from two parents. The crossover mask is generated randomly.

Mutation: The mutation operator randomly flips some of bits in a string. Mutation can occur at each bit position in a string with some small probability.

Reproduction: The reproduction operator clones an individual to create new one. In the case of elitism, best individuals are copied to the next generation to make sure that they are not destroyed by the crossover operation.

2.7 Genetic Programming

A system of Genetic Programming (GP) consists of individuals which correspond to computer programs. The computer programs are usually represented as trees. The hypothesis represented by these trees are more complex and more flexible than GA. Koza [81] showed that a wide variety of problems in many different fields can be expressed as the problems of searching for the optimal computer programs and can be solved by domain-independent operations. Koza [81] also argues that GP "actively encourages a diverse set of clearly inconsistent and contradictory approaches in attempting to solve a problem". Diversity is one of the key components in GP.

Each tree consists of internal nodes and leaf nodes. Internal nodes in a tree are "functional nodes" which represent functional operations such as AND, OR, $+$, and $-$ on the children nodes. The set of all functions is called the "function set". Leaf nodes which are called "terminal nodes" correspond to either "variable nodes" whose values are provided by input variables or "constant nodes" whose values are randomly generated.

The whole set of individuals is called a population and GP optimizes the population according to a fitness function. Algorithm 2.3 describes the main steps of GP (Banzhaf et al. [3], Neshatian, K. [114]).

In the standardized GP process, there are several options at each step of the process. We explain them referring to GPLAB (Silva and Almeida [142]).

Algorithm 2.3: Main Steps of GP

```
Initialization:
Creation of Initial Population
Fitness Evaluation
while (termination conditions are not satisfied) do
  Selection
  Genetic Operation (crossover, mutation, reproduction)
  Fitness Evaluation
  Environmental Selection
end while
```

Creation of Initial Population

The first step of the GP process is to create an initial population. There are three common ways of creating initial populations:

- Full: New individuals receive functional nodes until the specified tree depth is reached. Therefore the full capacity of the tree is utilized.
- Grow: New individuals receive randomly functional nodes or leaf nodes. Therefore individuals in an initial population have very different shapes.
- Ramped half-and-half: The half of new individuals are created using the Full method and the other half are created using the Grow method.

Selection

GP can be used with different selection methods. In roulette wheel selection, the probability of selecting an individual is given by the ratio of its fitness to the fitness of other members of the current population. GPLAB provides options with regards to the way of evaluating fitness either by its absolute fitness value or by its rank in the population. Tournament selection is also a popular selection method. In GPLAB, each parent is chosen by randomly drawing a number of individuals and selecting best of them.

Genetic Operation

Crossover: The crossover operation produces two new offspring from two parent trees. One node is randomly chosen in each parent tree. The subtrees rooted at the selected node are swapped to generate two offspring (Figure 2.7).

Mutation: The mutation operator randomly choose one node in a tree and replace the subtree rooted at the selected node with a randomly-created subtree.

Reproduction: The reproduction operator clones an individual to create new one.

Environmental Selection

After the genetic operation there are two sets of populations; parent and children. We have a choice to formulate the next generation among them. GPLAB provides the following options:

- Replace: The new children replace the parent population.
- Keepbest: The best individual from parents and the best individual from children are both selected, The remaining individuals are selected by Replace method.
- Halfelitism: The best half individuals from parents and the best half individuals from children are selected.
- Totalelitism: All the individuals are selected based on the fitness alone.

2.8 Bézier Curves and Surfaces

Bézier curves are developed as a tool to draw curves in blueprints. They were independently developed by P. de Casteljau and P. Bézier since the late 1950's. They are now standard tools for curve and surface description in CAD/CAM and computer graphics. In this section we provide a brief review of Bézier curve methods which are used in Chapter 3.

2.8.1 Bézier curves

Bézier curves are polynomial curves which are constructed by a sequence of linear interpolations. Let $\mathbf{b}_0$ and $\mathbf{b}_1$ be two distinct points in $\mathbb{R}^n$. The first degree Bézier curve $\mathbf{b}_0^1(t)$ is defined as a linear interpolation between the two points:

$$\mathbf{b}_0^1(t) = (1-t)\mathbf{b}_0 + t\mathbf{b}_1, \qquad t \geq 0, \quad t \in \mathbb{R}.$$

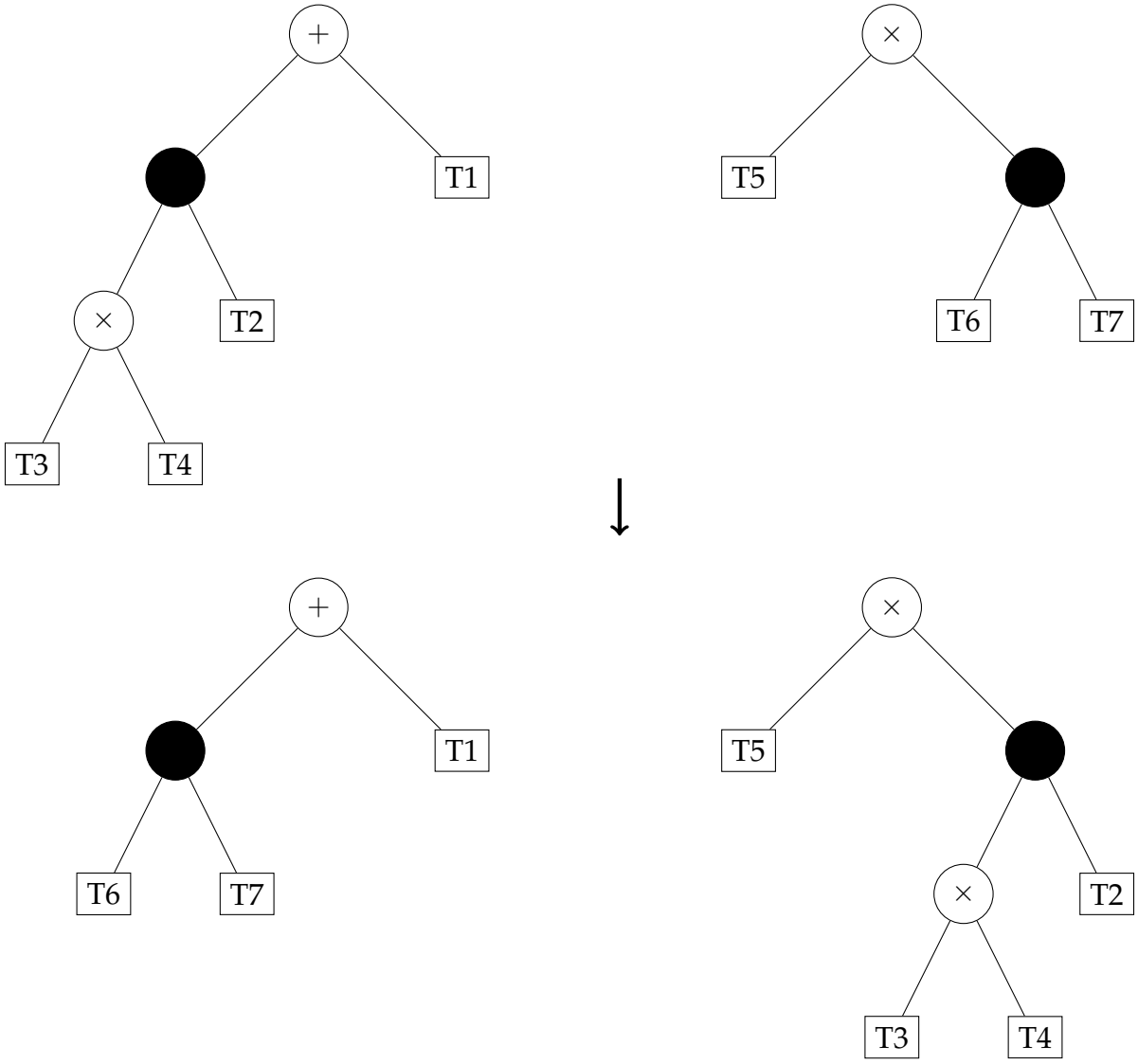


Figure 2.7: Crossover Operation

The second degree Bézier curve $\mathbf{b}_0^2(t)$ is defined on three points $\mathbf{b}_0$, $\mathbf{b}_1$ and $\mathbf{b}_2$:

$$\begin{aligned}
\mathbf{b}_0^1(t) &= (1-t)\mathbf{b}_0 + t\mathbf{b}_1, \qquad t \geq 0 \\
\mathbf{b}_1^1(t) &= (1-t)\mathbf{b}_1 + t\mathbf{b}_2, \qquad t \geq 0 \\
\mathbf{b}_0^2(t) &= (1-t)\mathbf{b}_0^1(t) + t\mathbf{b}_1^1(t), \\
&= (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2, \qquad t \geq 0
\end{aligned}$$

When t goes from 0 to 1, the Bézier curve $\mathbf{b}_0^2(t)$ moves from the point $\mathbf{b}_0$ to $\mathbf{b}_2$ as illustrated in Figure 2.8.

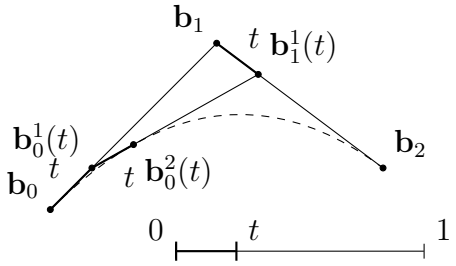


Figure 2.8: Second degree Bézier curve

In general the n-th degree Bézier curve $\mathbf{b}_0^n(t)$ is defined on $n+1$ points as follows:

$$\begin{aligned}
\mathbf{b}_0^n(t) &= (1-t)\mathbf{b}_0^{n-1}(t) + t\mathbf{b}_1^{n-1}(t), \\
&= \binom{n}{0}(1-t)^n\mathbf{b}_0 + \binom{n}{1}t(1-t)^{n-1}\mathbf{b}_1 +, \ldots, + \binom{n}{n}t^n\mathbf{b}_n, \qquad t \geq 0
\end{aligned}$$

where $\mathbf{b}_i^0 = \mathbf{b}_i$, for $i = 0, \ldots, n$.
The points $\mathbf{b}_0, \ldots, \mathbf{b}_n$ are called "Bézier points". $\binom{n}{i}t^i(1-t)^{n-i}$ are called "Bernstein polynomials" of degree n and denoted as $B_i^n(t)$:

$$B_i^n(t) = \binom{n}{i}t^i(1-t)^{n-i}, \qquad i = 0, 1, \ldots, n$$

In practice, the parameter transformations are frequently required.

$$u(t) = a(1-t) + bt, \qquad t \in [0,1], \quad a \neq b$$

Therefore, it is convenient to use the parameter u instead of t in the left hand side of the equation of the Bézier curve:

$$\mathbf{b}(u) = \mathbf{b}(u(t)) = \mathbf{b}_0^n(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t), \qquad t = \frac{u-a}{b-a} \tag{2.41}$$

u is called the "global parameter" and t is called the "local parameter" (Prautzsch et al. [122]).

The derivative of Bernstein polynomial of degree n is computed as:

$$\frac{d}{dt}B_i^n(t) = n\left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right), \qquad i = 0, \ldots, n.$$

Therefore the derivative of Bézier curve is

$$\frac{d}{dt}\mathbf{b}(u) = \frac{n}{b-a}\sum_{i=0}^{n-1}\Delta\mathbf{b}_i B_i^{n-1}(t), \qquad t = \frac{u-a}{b-a}$$

where $\Delta\mathbf{b}_i = \mathbf{b}_{i+1} - \mathbf{b}_i$.
The indefinite integral of Bézier curve is computed as:

$$\int \mathbf{b}(u)du = \sum_{i=0}^{n+1}\mathbf{c}_i B_i^{n+1}(t), \qquad t = \frac{u-a}{b-a}$$

where

$$\begin{aligned}\mathbf{c}_i &= \mathbf{c}_{i-1} + \frac{b-a}{n+1}\mathbf{b}_{i-1} \\ &= \mathbf{c}_0 + \frac{b-a}{n+1}\left(\mathbf{b}_0 + \cdots + \mathbf{b}_{i-1}\right).\end{aligned}$$

and $\mathbf{c}_0$ is an arbitrary integration constant (Prautzsch et al. [122]).
Especially the definite integral from a to b takes a simple form:

$$\int_a^b \mathbf{b}(u)du = \frac{b-a}{n+1}\sum_{i=0}^{n}\mathbf{b}_i.$$

So far we are dealing with the Bézier curves of parametric form. For instance, in two dimensional spaces the Bézier curves are written as:

$$\mathbf{b}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}.$$

Using the identity $\sum_{i=0}^{n}\frac{i}{n}B_i^n(t) = t$, Bézier curves can be expressed in a functional form $y = f(t)$:

$$\mathbf{b}(t) = \begin{bmatrix} t \\ f(t) \end{bmatrix}.$$

Expressing $f(t)$ in terms of the Bernstein polynomials:

$$f(t) = b_0 B_0^n(t) + \cdots + b_n B_n^n(t)$$

where $b_0, \ldots, b_n$ are scalar values, we can write the functional curve as:

$$\mathbf{b}(t) = \sum_{i=0}^{n} \begin{bmatrix} \frac{i}{n} \\ b_i \end{bmatrix} B_i^n(t).$$

For a general u,

$$\mathbf{b}(u) = \sum_{i=0}^{n} \begin{bmatrix} a + \frac{i(b-a)}{n} \\ b_i \end{bmatrix} B_i^n(t), \qquad t = \frac{u-a}{b-a}.$$

Using this functional form, the integral of $f(u)$ is computed as:

$$\int_a^b f(u)du = \frac{b-a}{n+1} \sum_{i=0}^{n} b_i.$$

2.8.2 Interpolation

Interpolation finds a curve that passes through given points. The cubic Hermite interpolation method is constructed as follows.
Given $m+1$ points $\mathbf{p}_0, \ldots, \mathbf{p}_m$ and the derivatives $\mathbf{d}_0, \ldots, \mathbf{d}_m$, there is a unique piecewise cubic C^1 curve $\mathbf{s}(u)$ over $[u_0, u_m]$ such that

$$\mathbf{s}(u_i) = \mathbf{p}_i, \qquad \mathbf{s}'(u_i) = \mathbf{d}_i.$$

The Bézier representation of the curve is written as:

$$\begin{aligned}
\mathbf{s}(u) &= \sum_{j=0}^{3} \mathbf{b}_{3i+j} B_j^3(t), \qquad t = \frac{u-u_i}{u_{i+1}-u_i}, \quad u \in [u_i, u_{i+1}], \quad i = 0, \ldots, m-1 \\
\mathbf{b}_{3i} &= \mathbf{p}_i \\
\mathbf{b}_{3i+1} &= \mathbf{p}_i + \mathbf{d}_i \frac{\Delta u_i}{3} \\
\mathbf{b}_{3i+2} &= \mathbf{p}_{i+1} - \mathbf{d}_{i+1} \frac{\Delta u_i}{3}
\end{aligned}$$

where $\Delta u_i = u_{i+1} - u_i$.
The derivatives $\mathbf{d}_i$ are estimated as:

$$\begin{aligned}
\mathbf{d}_i &= (1-\alpha_i) \frac{\Delta \mathbf{p}_{i-1}}{\Delta u_{i-1}} + \alpha_i \frac{\Delta \mathbf{p}_i}{\Delta u_i}, \qquad \text{for} \quad i = 1, \ldots, m-1 \\
\mathbf{d}_0 &= 2\frac{\Delta \mathbf{p}_0}{\Delta u_0} - \mathbf{d}_1 \\
\mathbf{d}_m &= 2\frac{\Delta \mathbf{p}_{m-1}}{\Delta u_{m-1}} - \mathbf{d}_{m-1}
\end{aligned}$$

where $\alpha_i = \frac{\Delta u_{i-1}}{\Delta u_{i-1} + \Delta u_i}$ and $\Delta \mathbf{p}_i = \mathbf{p}_{i+1} - \mathbf{p}_i$ (Prautzsch et al. [122]).

2.8.3 Tensor product surfaces

Tensor product surfaces are constructed by sliding a curve such that its Bézier points move along some curves (Prautzsch et al. [122]). Let's consider a Bézier curve of degree m:

$$\mathbf{b}(u) = \sum_{i=0}^{m} \mathbf{b}_i B_i^m(t), \qquad t = \frac{u-a}{b-a}.$$

Let each $\mathbf{b}_i$ traverse a Bézier curve of degree n:

$$\mathbf{b}_i = \mathbf{b}_i(v) = \sum_{j=0}^{n} \mathbf{b}_{i,j} B_j^n(s), \qquad s = \frac{v-c}{d-c}.$$

Combining those two equations we obtain the point $\mathbf{b}(u,v)$ on the surface:

$$\mathbf{b}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{b}_{i,j} B_i^m(t) B_j^n(s)$$

where u and v are global and t and s are local parameters.
This model works well if the Bézier points $\mathbf{b}_{i,j}$ locate regularly on each direction of the axis.
The derivative of $\mathbf{b}(u,v)$ is computed as:

$$\frac{\partial^{1+1}}{\partial u \partial v}\mathbf{b}(u,v) = \frac{mn}{(b-a)(d-c)} \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} \Delta^{1,1}\mathbf{b}_{i,j} B_i^{m-1}(t) B_j^{n-1}(s)$$

where $\Delta^{1,1}\mathbf{b}_{i,j} = \mathbf{b}_{i+1,j+1} - \mathbf{b}_{i+1,j} - \mathbf{b}_{i,j+1} + \mathbf{b}_{i,j}$.
A functional form of Bézier surface in three dimensional space is

$$\mathbf{b}(u,v) = \begin{bmatrix} u \\ v \\ f(u,v) \end{bmatrix} = \sum_{i=0}^{m}\sum_{j=0}^{n} \begin{bmatrix} a + \frac{i(b-a)}{m} \\ c + \frac{j(d-c)}{n} \\ b_{ij} \end{bmatrix} B_i^m(t) B_j^n(s), \quad t = \frac{u-a}{b-a}, \quad s = \frac{v-c}{d-c}.$$

The integral of $f(u,v)$ is computed by Fubini's theorem as follows:

$$\int_a^b \int_c^d f(u,v)dudv = \frac{(b-a)(d-c)}{(m+1)(n+1)} \sum_{i=0}^{m}\sum_{j=0}^{n} b_{ij}. \tag{2.42}$$

2.9 Gray Code

In this section we present a brief review of the basics of Gray code. In the literature, the binary representation is often used for the transformation of real solutions of benchmark functions. However, it does not preserve the relative distance between

the points in spaces. For instance, the integers $\{0, 1, 2, 3, 4, 5, 6, 7\}$ are represented as $\{000, 001, 010, 011, 100, 101, 110, 111\}$ in the binary system. The distance between 3 and 4 is one but the Hamming distance between 011 and 100 is three. Gray codes are minimal change orderings of bit strings in which successive strings differ by a single bit.

2.9.1 Minimal Change Ordering

We consider a particular class of Gray codes called the binary reflected Gray code. An n-bit Gray code $G(n)$ consists of 2^n n-bit strings $G_{n,0}, \ldots, G_{n,2^n-1}$.

$$G(n) = \begin{pmatrix} G_{n,0} \\ G_{n,1} \\ \vdots \\ G_{n,2^n-1} \end{pmatrix}$$

where

$$G_{n,k} = (g_{k,n-1}, \ldots, g_{k,0}), \quad g_{k,j} \in \{0, 1\} \quad \text{for } k = 0, \ldots, 2^n - 1,\ j = 0, \ldots, n - 1.$$

Starting with

$$G(1) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$G(n + 1)$ is constructed as

$$G(n+1) = \begin{pmatrix} 0G_{n,0} \\ 0G_{n,1} \\ \vdots \\ 0G_{n,2^n-1} \\ 1G_{n,2^n-1} \\ 1G_{n,2^n-2} \\ \vdots \\ 1G_{n,0} \end{pmatrix}$$

In the binary reflected Gray code, $G(n+1)$ consists of a copy of $G(n)$ with a "0" attached to each bit string and a copy of $G(n)$ in reverse order with a "1" attached to each string.

For instance,

$$G(2) = \begin{pmatrix} 00 \\ 01 \\ 11 \\ 10 \end{pmatrix}$$

$$G(3) = \begin{pmatrix} 000 \\ 001 \\ 011 \\ 010 \\ 110 \\ 111 \\ 101 \\ 100 \end{pmatrix}$$

A Gray code can be represented by the transition sequence — the ordered list of bit positions from right that change as we move from one string to the next (Reingold et al. [125]). For instance, the transition sequence T_3 of the above $G(3)$ is 1213121. Let the transition sequence of $G(n)$ be

$$T_n = t_1, \ldots, t_{2^n-1}.$$

From the above construction,

$$\begin{aligned} T_1 &= 1 \\ T_{n+1} &= T_n, n+1, T_n^{\text{Reversed}} \end{aligned}$$

Since $T_1^{\text{Reversed}} = T_1$, $T_k^{\text{Reversed}} = T_k$ for any $k > 0$. Therefore,

$$\begin{aligned} T_1 &= 1 \\ T_{n+1} &= T_n, n+1, T_n \end{aligned}$$

The same transition sequences can be also obtained by different recursive definition (Reingold et al. [125]) as follows:

$$\begin{aligned} T_1 &= 1 \\ T_{n+1} &= 1, t_1+1, 1, t_2+1, \ldots, 1, t_{2^n-1}+1, 1. \end{aligned} \tag{2.43}$$

We will use this representation below.

The binary representation $B(n)$ of n-bit strings for integers are written as

$$\begin{pmatrix} 0 \\ 1 \\ \vdots \\ 2^n - 1 \end{pmatrix} \iff B(n) = \begin{pmatrix} B_{n,0} \\ B_{n,1} \\ \vdots \\ B_{n,2^n-1} \end{pmatrix}$$

where

$$B_{n,k} = (b_{k,n-1}, \ldots, b_{k,0}), \quad b_{k,i} \in \{0,1\} \quad \text{for } k = 0, \ldots, 2^n - 1,\ i = 0, \ldots, n-1.$$

Starting with

$$B(1) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$B(n+1)$ is recursively defined as

$$\begin{pmatrix} 0 \\ 1 \\ \vdots \\ 2^{n+1} - 1 \end{pmatrix} \iff B(n+1) = \begin{pmatrix} B_{n,0}0 \\ B_{n,0}1 \\ B_{n,1}0 \\ B_{n,1}1 \\ \vdots \\ B_{n,2^n-1}0 \\ B_{n,2^n-1}1 \end{pmatrix} \tag{2.44}$$

For instance,

$$B(2) = \begin{pmatrix} 00 \\ 01 \\ 10 \\ 11 \end{pmatrix}$$

$$B(3) = \begin{pmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{pmatrix}$$

We can construct $G(n)$ from $B(n)$ and vice versa.

Lemma 2.6. *Suppose the bit string* $B_{n,k} = (b_{k,n-1}, \ldots, b_{k,0})$ *is expressed as* $B_{n,k} = (b_{k,n}, b_{k,n-1}, \ldots, b_{k,0})$ *with* $b_{k,n} = 0$*. Then,*

$$g_{k,j} = (b_{k,j} + b_{k,j+1}) \mod 2 \tag{2.45}$$

$$b_{k,j} = \sum_{i=j}^{n-1} g_{k,i} \mod 2 \tag{2.46}$$

$$\textit{for } \; k = 0, \ldots, 2^n - 1, \; j = 0, \ldots, n-1$$

Proof. We give a simple proof by induction on n. For $n = 1$,

$$B(1) = \begin{pmatrix} 00 \\ 01 \end{pmatrix}$$

with $b_{k,n} = 0$. By the operation (2.45), $G(1)$ is certainly reproduced as

$$G(1) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

.

Suppose that for $n = i$, $G(i)$ are reproduced from $B(i)$ by the operation (2.45). Let the transition sequence of $G(i)$ be

$$T_i = t_1, \ldots, t_{2^i - 1}.$$

From (2.44),

$$B(i+1) = \begin{pmatrix} B_{i,0}0 \\ B_{i,0}1 \\ B_{i,1}0 \\ B_{i,1}1 \\ \vdots \\ B_{i,2^i-1}0 \\ B_{i,2^i-1}1 \end{pmatrix}$$

By the operation (2.45), $B_{i,k}0$ and $B_{i,k}1$ are transformed to $G_{i,k}0$ and $G_{i,k}1$ (if $b_{k,0}$ (the first bit of $B_{i,k}$ from right) $= 0$) or $G_{i,k}1$ and $G_{i,k}0$ (if $b_{k,0} = 1$), respectively. $B_{i,k}1$ and $B_{i,k+1}0$ are transformed to $G_{i,k}0$ and $G_{i,k+1}0$ (if $b_{k,0} = 1$) or $G_{i,k}1$ and $G_{i,k+1}1$ (if $b_{k,0} = 0$), since $b_{k,0}+1 = b_{k+1,0}$ (the first bit of $B_{i,k+1}$ from right) ($\mod 2$). The transition position from right between $G_{i,k}0$ and $G_{i,k}1$ or $G_{i,k}1$ and $G_{i,k}0$ is one and the transition position between $G_{i,k}0$ and $G_{i,k+1}0$ or $G_{i,k}1$ and $G_{i,k+1}1$ is t_k+1. Therefore by the operation (2.45) the transition sequence $1, t_1 + 1, 1, t_2 + 1, \ldots, 1, t_{2^i-1} + 1, 1$ is generated. From (2.43), it

is the same transition sequence of $G(i+1)$. By the operation (2.45), $B_{i,0}0 = (0,\ldots,0)$ is obviously mapped to $G_{i,0}0 = (0,\ldots,0)$. Started with the bit string $G_{i,0}0 = (0,\ldots,0)$, the above transition sequence reproduces $G(i+1)$.

For the proof of (2.46), we take the summation of (2.45).

$$\begin{aligned}\sum_{i=j}^{n-1} g_{k,i} &= \sum_{i=j}^{n-1} (b_{k,i} + b_{k,i+1}) \mod 2 \\ &= (b_{k,j} + b_{k,n}) \mod 2 \\ &= b_{k,j} \mod 2\end{aligned}$$

since $b_{k,n} = 0$.

□

2.10 Literature Review

In this section we provide a literature review for the hyper-parameter search and the feature selection in the machine learning fields and the evolutionary computation fields.

2.10.1 Hyper-parameter Search

Hyper-parameters are parameters which are specified before the learning algorithm starts to solve a problem. Learning algorithms in machine learning fields typically contain some hyper-parameters.

Hyper-parameter Search by Grid Search and Cross-validation Methods

The most traditional way of performing hyper-parameter search is the grid search, which is an exhaustive searching through a specified subset of the hyper-parameter space. It is often combined with cross-validation to maximally exploit the information in the training set. If we use a finer calibration of the grid step, the grid search more exhaustively explores the search spaces but it becomes a more time-consuming process. Yao et al. [179] proposed the multilevel grid search which starts with a rough calibration of grid step and in the subsequent round both the grid step and the search region are halved around the optimal values of hyper-parameters in the current round. The search region is halved in the same way as the binary search. As for the cross-validation, Kohavi [79] examined the effect of the number of folds in cross-validation

methods. If we choose a larger number of folds, we would obtain the estimates which are almost unbiased but have larger variances, which lead to unreliable estimations. It is a bias-complexity trade-off discussed in (Shalev-Shwartz and Ben-David [137]). Kohavi [79] recommends ten-fold cross-validation even if computing power allows us to use more folds.

Hyper-parameter Search by Bayesian Optimization Methods

Bayesian statistics treat an event as a random function and put a prior distribution (or prior belief) over it. After gathering observation, the prior distribution is updated to form the posterior distribution through the Bayesian theorem. This mechanism is applied to hyper-parameter search.

On a minimization problem of a function $f(\mathbf{x})$, Bayesian optimization constructs a probabilistic model for $f(\mathbf{x})$. When performing the Bayesian optimization, we need to select a "prior" over the function. The "Gaussian Process" (GP) prior is commonly chosen due to its tractability. In GP, any finite set of $\mathbf{x}$ induces a multivariate Gaussian distribution. We also need to select an "acquisition function" $\mathcal{S}$ which determines the next point of $\mathbf{x}$ (Snoek et al. [143]).

Let $\mathcal{H}$ be a observation history. The work flow of GP is described using Sequential Model-Based Global Optimization (SMBO) algorithm. The point of this algorithm is to minimize the acquisition function $\mathcal{S}$ instead of f as shown in Algorithm 2.4 (Bergstra et al. [10]).

Algorithm 2.4: SMBO for GP

Require: $\mathcal{S}$, $\mathcal{H}$, T
  $\mathcal{H} \leftarrow \emptyset$
  for $t \leftarrow 1$ to T **do**
    $\mathbf{x}^* \leftarrow \mathrm{argmax}_{\mathbf{x}}\, \mathcal{S}(\mathbf{x}, \mathcal{H})$
    Evaluate $f(\mathbf{x}^*)$
    $\mathcal{H} \leftarrow \mathcal{H} \bigcup (\mathbf{x}^*, f(\mathbf{x}^*))$
  end for
  Return $\mathcal{H}$

Let's denote $\mathbf{x}_{best} = \mathrm{argmin}_{\mathbf{x} \in \mathcal{H}}\, f(\mathbf{x})$. and the cumulative distribution function of standard normal distribution as $\Phi(\mathbf{x})$. An acquisition function, Probability of Improve-

ment (PI) is defined as

$$\mathcal{S}_{PI}(\mathbf{x}, \mathcal{H}) := \Phi\left(\frac{f(\mathbf{x}_{best}) - \mu(\mathbf{x}, \mathcal{H})}{\sigma(\mathbf{x}, \mathcal{H})}\right).$$

where $\mu(\mathbf{x}, \mathcal{H})$ and $\sigma(\mathbf{x}, \mathcal{H})$ are an predicted mean and a standard deviation, respectively. $\mathrm{argmax}_{\mathbf{x}}\, \mathcal{S}_{PI}(\mathbf{x}, \mathcal{H})$ in Algorithm 2.4 means that we select $\mathbf{x}$ so that the difference $f(\mathbf{x}_{best}) - \mu(\mathbf{x}, \mathcal{H})$ is as small as possible.
Expected Improvement (EI) is the most commonly-used acquisition function:

$$\mathcal{S}_{EI}(\mathbf{x}, \mathcal{H}) := \mathbb{E}\left[\max\left(f(\mathbf{x}_{best}) - f(\mathbf{x}), 0\right)\right]$$

$\mathbf{x}^*\ (= \mathrm{argmax}_{\mathbf{x}}\, \mathcal{S}_{EI}(\mathbf{x}, \mathcal{H}))$ is computed using various methods such as an exhaustive grid search, Estimation of Distribution EDA [86], and Covariance matrix Adaptation - Evolutionary Strategy CMA-ES [60] (Bergstra et al. [10]).

Li et al [88] proposed a multi-arm bandit strategy called Hyperband which adaptively allocates more resources to randomly sampled hyper-parameter specifications based on the performance on sample data. It extends Successive Halving algorithm [145] which starts with uniformly allocating resources to a set of hyper-parameter specifications, evaluate the performance, throw the worst half and repeat until one specification remains. In their experiments Hyperband outperformed well-established Bayesian optimization algorithms.

Hyper-parameter Search by PSO

In PSO, each particle has a memory of the best performance by itself and by the swarm (or the neighborhoods). Particles are attracted to those best points and the areas around them are intensively explored. The intensive local search ability of PSO has been utilized for the hyper-parameter search in SVMs and new methods are proposed to enhance the ability. Lin et al. [92] use PSO for the combined task of hyper-parameter search and feature selection. Sherin and Supriya [100] apply the BAT algorithm [176], a meta-heuristic method inspired by the echolocation mechanism of bats, to hyper-parameter search, and compare the results with the PSO approach. Yang et al. [175] incorporate a chaos operator in PSO to enhance the convergence speed and precision of the hyper-parameter search.

Bao et al. [4] conduct hyper-parameter search experiments using the framework of Memetic Algorithms (MAs), which is inspired by the Darwinian principles of natural evolution and the Dawkins notion of a meme (Moscato and Norman [113], Moscato [112]). The meme is defined as "a unit of cultural evolution that is capable of local

refinement" (Bao et al. [4]). The MAs aim at the optimal balance between exploration and exploitation of the search space. Bao et al. [4] use PSO to explore the space and pattern search to conduct the exploitation of the space. In the two dimensional experiments with Gaussian kernel a five point unit-size cross (+) pattern is employed.

$$P = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \end{bmatrix} \tag{2.47}$$

For a point $\mathbf{x}_0$, the points in the neighbors Ω are evaluated

$$\Omega = \{\mathbf{x}_0 + \Delta p_k | \text{for each column } p_k \text{ in } P\}$$

where Δ is a search step. If there are improvements in Ω, $\mathbf{x}_0$ is replaced by the best point. Otherwise, Δ is decreased to $\frac{\Delta}{2}$. The pattern search is continued until the original search step Δ_0 is reduced to $\frac{\Delta_0}{8}$. The experiments compare PSO without local search with four patterns of selection strategy:

- MA1: all newly generated particles are refined with the pattern search;
- MA2: The pattern search is applied to each newly generated particle with predefined probability 0.1;
- MA3: The pattern search is applied to the two best fitness particles;
- MA4: The pattern search is applied to each newly generated particle with probability proportion to its fitness;

The maximum number of evaluation of the particles is fixed to 1500. MA4 achieves the best result and all four MA achieve better accuracy then PSO without local search.

2.10.2 Feature Selection

Feature selection methods can be organized into three categories: *filter* methods, *wrapper* methods and *embedded* methods. Filter methods evaluate the relevance among input variables and output labels independent of the classification algorithms. Common choices for this analysis are simple univariate statistics such as Pearson's correlation coefficient, Wilcoxon statistics and t-statistics (Guyon and Elisseeff [58]).

In wrapper methods, a classification learning algorithm is "wrapped" in the search process. During the search, a candidate subset of features is evaluated by projecting the input data onto the selected features, training a classifier using these features, and

then the performance of the trained classifier on the validation set is interpreted as the fitness of the candidate subset of features.

In embedded methods, the search of optimal subsets of features is embedded in the classification algorithm. Embedded methods have the advantage that they interact with the classification algorithms, but they are less computationally intensive than wrapper methods (Saeys et al. [131]).

Feature Selection by Traditional Methods

This section review traditional feature selection methods. Some wrapper methods will be used in the experiments in the thesis.

Wrapper Approach Sequential forward selection (SFS) methods and sequential backward selection (SBS) methods are two commonly-used wrapper algorithms. SFS starts with an empty set of features. We denote the feature subset $\mathcal{F}$. After testing the addition of each feature, SFS adds the feature whose inclusion gives the largest improvement to $\mathcal{F}$. This process is repeated until the addition of a feature does not improve the fitness of $\mathcal{F}$. On the other hand, SBS starts with a set $\mathcal{F}$ of all features. After testing the elimination of each feature, SBS removes the feature whose elimination gives the largest improvement to $\mathcal{F}$. This process is repeated until the elimination of a feature does not improve the fitness of $\mathcal{F}$. Both methods suffer from the problem that once a feature is added (removed), it can not be removed (added), which is called a "nesting effect" (Pudil et al. [123]).

To overcome this problem, Stearns [147] proposed the "plus-l-minus-r" (l-r) method, which applies the SFS l times and then the SBS r times. However, there is no theoretical way to determine the values of l and r to obtain the best feature subset. Pudil et al. proposed Sequential Forward Floating Selection (SFFS) and Sequential Backward Floating Selection (SBFS) to determine the values of l and r automatically and adaptively. In the case of SFFS, after the inclusion process of SFS, We select the least significant feature among the current set of features except for the last added feature. If the elimination of the (least significant) feature leads to the improvement over the fitness of the subset of features excluding the last feature , we eliminate the feature from the subset and continue the elimination process.

Filter Approach The basic idea of filter-based feature selection is to select a feature which has a maximum relevance to the target variable and minimum relevance (re-

dundancy) among the selected features. Correlation in statistics (Yu and Liu [177]) and mutual information are commonly used to measure the relevance. The mutual information is defined using a concept of entropy, which is a measure of an uncertain (random) variable. Let X, Y be two discrete random variables and $\mathbf{x} \in \Omega$, $\mathbf{y} \in \Delta$ be the realization of X and Y. Let $p(\mathbf{x}), \mathbf{x} \in \Omega$ be the probability distribution of $\mathbf{x}$. An entropy of $\mathbf{x}$ is defined as:

$$H(X) := -\sum_{\mathbf{x}\in\Omega} p(\mathbf{x}) \log p(\mathbf{x}).$$

Given a joint distribution $p(\mathbf{x}, \mathbf{y})$, the conditional entropy is defined as:

$$H(X \mid Y) := -\sum_{\mathbf{x}\in\Omega} \sum_{\mathbf{y}\in\Delta} p(\mathbf{x}, \mathbf{y}) \log p(\mathbf{x} \mid \mathbf{y})$$

The mutual information is defined as:

$$\begin{aligned} I(X, Y) &:= -\sum_{\mathbf{x}\in\Omega} \sum_{\mathbf{y}\in\Delta} p(\mathbf{x}, \mathbf{y}) \log \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})p(\mathbf{y})} \\ &:= H(X) - H(X \mid Y) \end{aligned}$$

The mutual information is a measure of the amount of information that one random variable possesses about another random variable. If $I(X, Y)$ is large, the two variables X and Y are closely related.

Let A be an initial set of n features and F be a set of selected features. Let O be a target variable and $\mathcal{S}$ be a fitness evaluation function.

The process of the feature selection starts by selecting a feature which has a maximum relevance to the target variable O. It continues by selecting a best feature from A with respect to an evaluation function $\mathcal{S}$ (Algorithm 2.5).

Algorithm 2.5: Feature Selection

Require: $\mathcal{S}$, A, $F = \emptyset$, O

Compute $I(O, f_i)$ for $\forall f_i \in A$

Select $i^* = \text{argmax}_{i\in A}\, I(O, f_i)$, $A = A \setminus \{f_{i^*}\}$, $F = \{f_{i^*}\}$

while (termination conditions are not satisfied) **do**

Select $i^* = \text{argmax}_{i\in A}\, \mathcal{S}(i)$, $A = A \setminus \{f_{i^*}\}$, $F = F \bigcup \{f_{i^*}\}$

end while

Return F

As the evaluation function $\mathcal{S}$, Battiti [7] proposed the following model which is called MIFS:

$$\mathcal{S}(i) := I(O, f_i) - \beta \sum_{f_s \in F} I(f_s, f_i)$$

where β is a user-defined constant which balances the relevant measure $I(O, f_i)$ and the redundancy measure $\sum_{f_s \in F} I(f_s, f_i)$.
Kwak and Choi [84] noticed that the MIFS method does not work well in nonlinear problems, and proposed the following model which is called MIFS-U:

$$\mathcal{S}(i) := I(O, f_i) - \beta \sum_{f_s \in F} \frac{I(O, f_i)}{H(f_s)} I(f_s, f_i).$$

Peng et al. [119] proposed the following model called mRMR:

$$\mathcal{S}(i) := I(O, f_i) - \frac{1}{|F|} \sum_{f_s \in F} I(f_s, f_i).$$

This model does not include the constant β whose optimal values are usually difficult to find.
Foithong et al. [46] proposed the following model which quantifies the dependency among features with respect to the target variable:

$$\mathcal{S}(i) := I(O, f_i) - \frac{H(f_i \mid O)}{H(f_i)} \sum_{f_s \in F} \frac{I(f_s, f_i) I(O, f_s)}{H(f_s) H(O)}.$$

Feature Selection by Evolutionary Computation

Feature (subset) selections by PSO are filter methods and wrapper methods and both binary PSO and continuous PSO are used for feature selection (Xue et al. [169]). The filter methods evaluate the relevance among input variables and output labels using some correlation measures such as rough sets theory, mutual information and entropy (Tran et al. [156]).

When continuous PSO is used for feature selection, a threshold ν is usually used to map the real values to $\{0, 1\}$ so that if the real values are larger than ν they are mapped to 1. Tran et al. [157] propose a entropy-based discretization scheme. The threshold ν for each feature is determined so that the divided sub-intervals return the highest information gain. In their model, the threshold ν is chosen from the set of "potential' thresholds which are determined by eliminating the lower values of information gain. Xue et al. [170] firstly introduced the multi-objective scheme (prediction accuracy + number of features) to feature selection in PSO, in which the gbest of each particle is set as one of the highest ranked nondominated solutions.

Feature (subset) selections by GP are filter methods and wrapper methods (Xue et al. [169]). GP has been used for feature selection in a variety ways. Suárez et al. [152] use two GP runs to first identify promising subsets of features and then use them in a classification problem. Gray et al. [55] use GP to evolve a classifier. The features that are actively used in the resulting classifier identify good predictors, and therefore, implicitly lead to feature selection. Bhowan et al. [12] use a similar mechanism in order to evolve ranking functions and achieve comparable performance with fewer numbers of features.

Feature Selection by SVMs

Feature (subset) selections by SVMs are embedded methods in which features are ranked according to their weights $\mathbf{w}$ in (2.48).

$$\min \left(\Omega(\mathbf{w}) + C \sum_{i=1}^{n} \ell \left(\mathbf{w}, \Phi(\mathbf{x}_i), y_i \right) \right) \quad , \tag{2.48}$$

In the case of linear models ($\Phi(\mathbf{x}_i) = \mathbf{x}_i$), $\mathbf{x}_i$ with small $\mathbf{w}_i$ does not have much influence for the determination of the optimal hyperplane. Hence the norm of weight can be used as the criteria for feature ranking (Guyon et al. [59]). If we use an ℓ_1-norm for the regularization function ($\Omega(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|$), the optimization problems (2.48) return the sparse solutions (Tibshirani [155]). It is known that the ℓ_2-norm shrinks the coefficients of correlated variables evenly. In the extreme case of d identical variables, we get identical coefficients of $\frac{1}{d}$. On the other hand the ℓ_1-norm "picks one and ignores the rest". (Friedman et al., 2010 [48])

In the case of nonlinear models, the coefficients $\boldsymbol{\theta}$ of the linear combination of kernels in (6.3) are used as the criteria for the feature ranking. A set of kernels is prepared so that each kernel corresponds to a single feature. Then the information about the relative importance of features is obtained by solving the MKL. As in the linear case, ℓ_1-norm MKL in (6.3) also returns sparse solutions with respect to the coefficients $\boldsymbol{\theta}$ of the kernel (Kloft [76]).

A linear SVM solver LIBLINEAR (Fan et al. [44]) conducts the feature selection using the following methods:

- Coordinate descent method using one dimensional Newton directions (CDN): (Yuan et al. [178])
  CDN uses the ℓ_1-norm and a squared hinge loss:

$$\ell \left(\mathbf{w}, \mathbf{x}_i, y_i \right) = \max(0, 1 - y(\langle \mathbf{w}, \mathbf{x} \rangle + b))^2$$

and solves (2.48) by the coordinate descent method.

- Generalized linear model with elastic net (GLMNET): (Friedman et al., 2010 [48]) GLMNET uses the squared hinge loss and an elastic-net penalty which is a combination of ℓ_2-norm and ℓ_1-norm. It also uses a coordinate descent method but its implementation is different from that of CDN.

Tan et al. [153] conducted a large-scale feature selection experiment using the ℓ_2-norm and the squared hinge loss with an additional constraint: $\boldsymbol{\eta} \circ \mathbf{x}$ is used instead of $\mathbf{x}$ where $\circ$ is a pair-wise product; and

$$\sum_{j=1}^{d} \eta_j \leq B, \quad \eta_j \in [0, 1], \quad j = 1, \ldots, d$$

where B is a positive integer which determines the number of features in the solutions. They adopted the accelerated proximal gradient method to efficiently solve the problem. Proximal algorithms were invented in 1970s (Rockafellar [129], [128]) and recently have become very popular tools for linear models (Boyd et al. [18], Parikh and Boyd [118]). They bring many advantages to convex optimization problems such as strong convexity and continuity (Bauschke and Combettes [8]). Liu et al. [94] apply ADMM (an Alternating Direction Method of Multipliers) to SVMs with the elastic-net penalty to conduct a large-scale feature selection. Balamurugan et al. [2] also use ADMM with an elastic-net based SVMs to classify structured objects (images, graphs, trees and sequences).

Part II

Hyper-parameter Search in SVMs

Chapter 3

Hyper-parameter Search using Bézier Curve Methods

In this chapter we propose surface estimation methods to find the optimal specification of hyper-parameters in Support Vector Machines (SVMs). Based on the samples in the hyper-parameter spaces we approximate the performance surface using Bézier curve methods. This geometrical approach allows us to use the information provided by the surface and find optimal specification of hyper-parameters in an automatic way.

3.1 Introduction

Successful applications of SVMs often depend on the values of hyper-parameters, which must be specified before solving the given problems. In order to find the optimal hyper-parameters, the data is typically divided into three parts: training, validation, and test sets. The prediction accuracy for each specification of the hyper-parameters is computed using a pair of the training set and the validation set. This is a simple mechanism to prevent SVMs from over-fitting the training data. The theoretical foundation for this validation approach is provided by Shalev-Shwartz and Ben-David [137].

In this chapter, we examine an idea of approximating the prediction accuracy by constructing the interpolating surface. We will construct the simplest form of surface called the cubic Hermite tensor product Bézier surface (Section 2.8). Based on the constructed surface, we can estimate the prediction accuracy at any points in the search spaces and identify the most relevant region for the finer search. In the experiments we will verify that the proposed method improves the efficiency over the more exhaustive

grid search without degrading the accuracy.

We measure the improvement of efficiency based on the number of SVM runs instead of the actual running time, because actual running time of each SVM varies substantially according to the various factors such as the size of training data, datasets, types of kernels.

Gap or weakness in previous research: The most common method for hyper-parameter search in SVMs is the grid search. The grid search method has the advantage that by using finer grids (more evaluation points), it approaches the exhaustive search. Therefore one can choose how arbitrarily accurate the grid search has to be. However, the computational complexity of the grid search grows exponentially with respect to the dimensionality of the hyper-parameter space.

Objectives of Chapter 3: The goal is to develop a hyper-parameter search algorithm that is more efficient than the grid search but does not degrade the accuracy.

3.2 Related Work

We start with a brief review of relevant concepts and previous works in the literature.

3.2.1 Surface of Prediction Accuracy for Gaussian Kernel

Although the surfaces for prediction accuracy in hyper-parameter spaces vary from data to data, the surfaces for each type of kernel have a characteristic configuration. For the Gaussian kernel it is known that the graph of the prediction accuracy for the Gaussian kernel parameter σ^2 and the regularization parameter C has a specific pattern. Keerthi and Lin [69] proved that when one parameter approaches 0 or ∞, while fixing the other parameter, the SVMs under-fit or over-fit the data. Therefore, there exist sector-shaped boundary curves which separate the "good" region from the under-fitting and over-fitting regions. Yan et al. [182] observed that usually there exist "optimal" regions within the "good" regions as depicted in Figure 3.1. These types of prior knowledge are quite useful to search for the optimal specification of hyper-parameters, especially to determine the range of the search region.

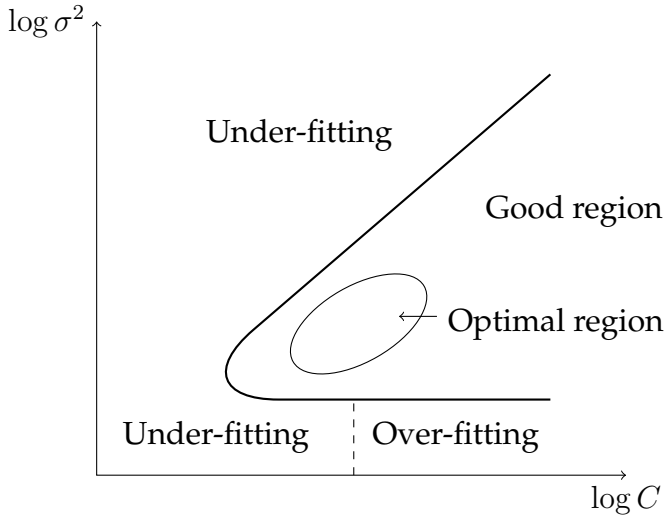


Figure 3.1: Good and Optimal regions

3.2.2 Surface Estimation Methods

In this chapter we apply surface estimation methods to hyper-parameter search in SVMs. Although it seems that there is no previous research similar to our attempt, the surface estimation methods are extensively studied and applied in several fields of science. For instance, contour mappings are drawn based on samples randomly taken in the region. In such a case, a standard approach is to divide the whole region into triangular regions and construct Bézier surfaces on the triangular patches (Farin [45]). The process of dividing the region into triangles is called triangulation. A common choice of the triangulation is called a Delaunay triangulation which avoids thin triangles wherever possible (Griffiths [57]). The drawback of this approach is that it is difficult to extend the triangulation in higher dimensional spaces. An another approach based on random samples is the interpolation method using radial basis functions (Liu [95], Wendland [166]). This method can be applied in higher dimensional spaces. However, it can not express the bumps and the valleys in the local region. In Bézier curve methods the sampling points are used as the points on the surface and the surface is constructed by interpolating those sample points. Samples on regular grids allows us to construct tensor product surfaces in spaces of any dimension. In the following sections we will construct cubic Hermite tensor product Bézier surfaces which interpolate surfaces by adding two Bézier points between adjacent sampling points along each axis (Section 2.8).

3.3 Proposed Methods

In this section we describe the surface estimation method for the optimal hyper-parameter search using Bézier Curve methods (Section 2.8.1).

3.3.1 Surface Estimation Methods by Bézier Curve (SEB)

One of the most common methods of hyper-parameter search is the grid search which is a simple way to exhaustively explore the whole space. We adopt a two-stage grid search as a reasonable strategy to optimally allocate the computational resources. It consists of a rough search in the broad region and a fine search in the restricted region which is determined by the first search. The second search can be conducted around the best point in the first search or in the restricted region proposed in Section 2.10.1. However, we propose a more intelligent way to identify the most relevant region. We call the surface estimation methods using Bézier Curve SEB in this thesis.

Cubic Hermite Tensor Product Bézier Surface

Sampling on the regular grids allows us to construct a cubic Hermite tensor product surface. The surface can be used to estimate the prediction accuracy at any points in the search region and identify the most relevant regions in the input domain. Let's denote the hyper-parameter spaces as S. For instance, if we use Gaussian kernels, a pair of the hyper-parameters is written as $\mathbf{x} = (C, \sigma^2) \in S \subset \mathbb{R}^2$. Each specification of hyper-parameters is evaluated by the prediction accuracy measured by validation datasets. Therefore we consider a mapping $z = f(\mathbf{x})$ where z represents the accuracy for the hyper-parameters $\mathbf{x} \in S$.

α-percent Region

We define α-percent region to quantify the volume under the surface. Suppose that we cut the z axis by a perpendicular plane at $z = b$. When the point b is high enough, there is no intersection between the surface and the plane. As the point b goes down, at some point the surface and the plane meet together. If the point b goes down further, the intersection between the surface and the plane becomes larger. We compute the volume under the surface corresponding to the set of domain $\{\mathbf{x} \mid f(\mathbf{x}) \geq b\}$. If the volume occupies the α percent of the whole volume, we call the set of domain as the α-percent region (Figure 3.2).

Definition 3.1. *α-percent region (Harlow et al. [61]):*

If the perpendicular plane at $z = b$ intersects the surface, then corresponding to the intersection, we identify the bounded domain $\mathbf{Dom}(b) = \{\mathbf{x} \mid f(\mathbf{x}) \geq b\}$. If the volume under the surface on the domain, $\int_{\mathbf{x} \in \mathbf{Dom}(b)} f(\mathbf{x}) d\mathbf{x}$, occupies the α percent of the whole volume under the surface, we call the domain $\mathbf{Dom}(b)$ as the α-percent region.

If a surface has large plateau (flat summit), it may not be possible to obtain the exact α-percent region. In the following experiments we will take a larger α-percent region in that case.

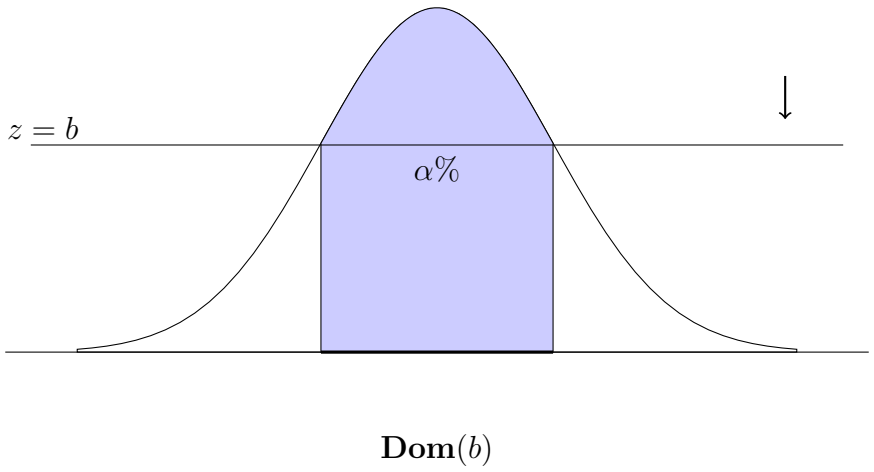


Figure 3.2: α-percent region

Work Flow of SEB

We adopt the two-stage sampling scheme to optimally use the computational resources. The proposed hyper-parameter search proceeds in the following steps:
Work flow of SEB:

1. Determination of the search region
   We decide the search region to cover the wide area of hyper-parameter spaces. In the case of Gaussian kernel in a two-dimensional space we choose the search region so as to include the "good" region of the sector shape (Section 3.2.1). In higher dimensional spaces it is helpful to visualize the α-percent region in three dimensional spaces by choosing three parameters and fixing others.

2. The first sampling and estimation:
   We conduct a first sampling on a rough grid in the entire region. Based on the

sampling points we construct a cubic Hermite tensor product Bézier surface by adding two Bézier points between the adjacent sample points along each axis. In this way, the input space is divided into cells whose vertices are sample points or Bézier points. We compute the volume under the surface by adding the volume under the surface on each cell and identify the α-percent region.

3. The second sampling and estimation:
   We pick up the cells which are entirely included in the α-percent region and replace the added Bézier points by actual sampling points which are obtained by running SVMs. Then we choose the best sampling point in the α-percent region as our estimation of the optimal hyper-parameters.

3.4 Experiments

The main purpose of the experiments is to examine the prediction accuracy of the proposed method by comparing the performance with the more exhaustive grid search. This section consists of three parts. In the first part we examine the search region. Then in the second part we conduct main experiments. Finally we examine the effects of grid size and the size α of α-percent region in terms of accuracy and efficiency.

3.4.1 Search Region

As stated in the previous section, the first step of hyper-parameter search is to determine the search region. A good starting point is to sample data from a wide region, for instance from a range $\{10^{-5}, \ldots, 10^{5}\}$. Then we run the SVM with this sample data. It would be useful to visualize the surface of the prediction accuracy. For two dimensional spaces a contour map can help the visualization of the surface. For three dimensional spaces we make use of the α-percent region in Section 3.3.1 for the visualization of the surface. Since the α-percent region consists of the points which have the high prediction accuracies, we can visualize the most important domain by drawing the α-percent region. For four dimensional spaces we draw the α-percent region in three dimensional spaces by choosing three parameters and fixing one parameter. Throughout the thesis we will use 17 benchmark datasets in Section 1.6 and two types of kernels in Section 2.3.3. We draw the surface of prediction accuracy for each dataset and for each kernel.

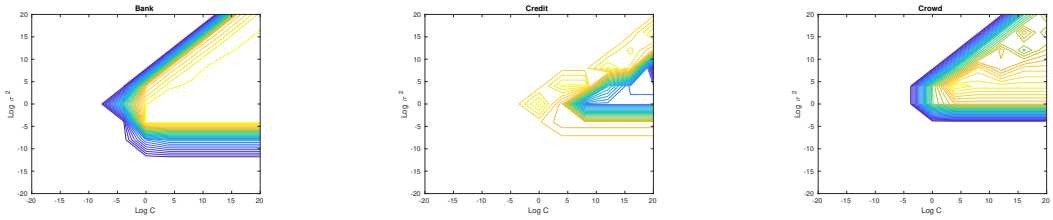


Figure 3.3: Contour Map for Banknote, Credit and Crowd Dataset (Gaussian Kernel)

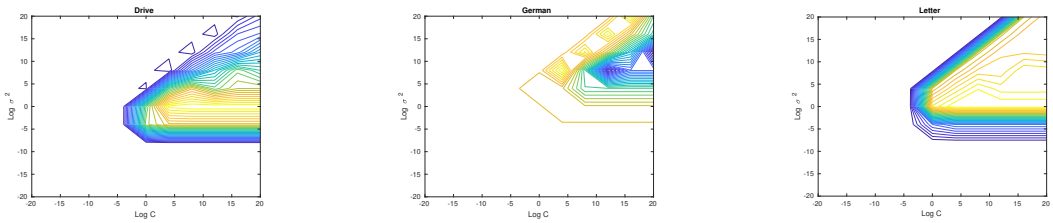


Figure 3.4: Contour Map for Drive, German and Letter Dataset (Gaussian Kernel)

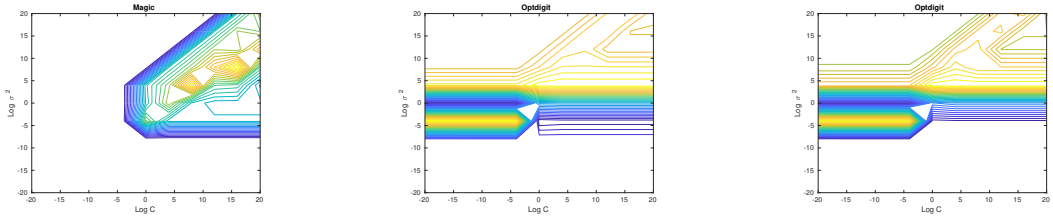
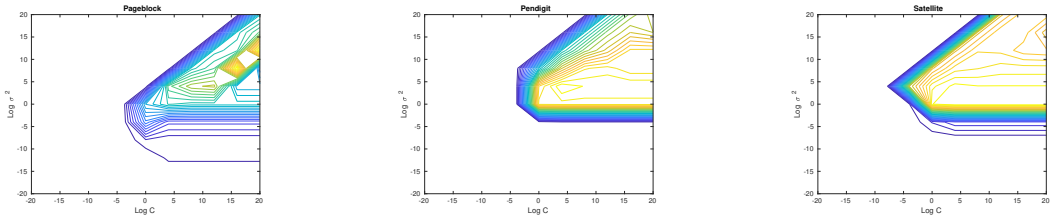


Figure 3.5: Contour Map for Magic, Occupancy and Optdigit Dataset (Gaussian Kernel)



-

in terms of t

Figure 3.6: Contour Map for Pageblock, Pendigit and Satellite Dataset (Gaussian Kernel)

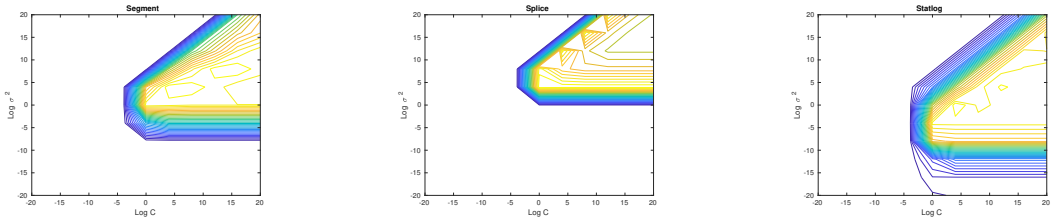


Figure 3.7: Contour Map for Segment, Splice and Statlog Dataset (Gaussian Kernel)

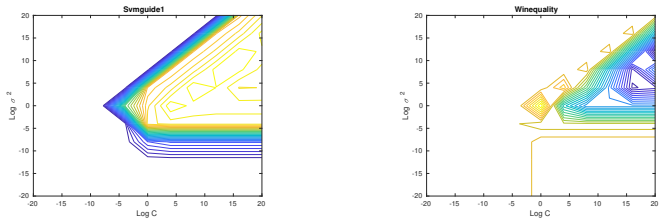


Figure 3.8: Contour Map for Svmguide1 and Winequality Dataset (Gaussian Kernel)

Figure 3.3 - 3.8 are the contour maps of σ^2 of the Gaussian kernel and the regularization parameter C. We draw those figures in the range of $\log_2 \sigma^2 \in [-20, 20]$ and $\log_2 C \in [-20, 20]$. They show the typical sector shape pattern of Gaussian kernel. Comparing those figures we set the search range of the parameters to $\log_2 \sigma^2 \in [-8, 10]$ and $\log_2 C \in [-8, 10]$.

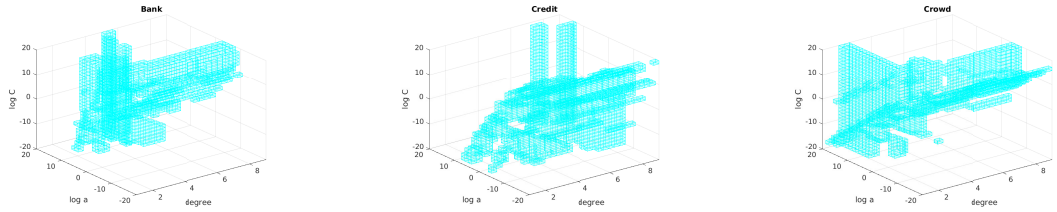


Figure 3.9: 10-percent region for Banknote, Credit and Crowd Dataset (Polynomial Kernel with $b = 1$)

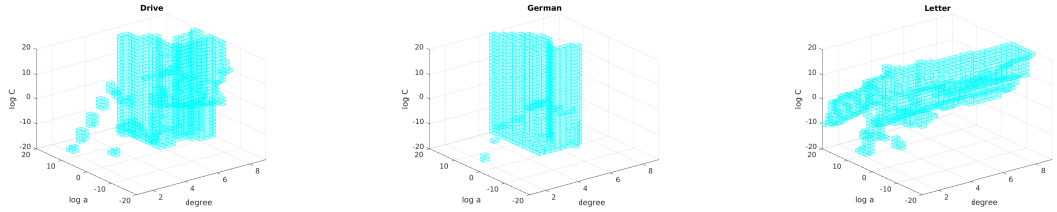


Figure 3.10: 10-percent region for Drive, German and Letter Dataset (Polynomial Kernel with $b = 1$)

Figure 3.9 - 3.14 are the 10-percent regions of polynomial kernel with $b = 1$. We draw the figure in the range of $\log_2 C \in [-20, 20]$, $\log_2 a \in [-20, 20]$ and $c \in [1, 9]$. We notice several characteristics of those figures:

- For $\log_2 a$, the positive range is more important for most datasets.
- For c (degree), the smaller values (the values close to 1) are more important.

Figure 3.15 - 3.20 are the 10-percent regions of polynomial kernel with the degree $= 3$. We draw the figure in the range of $\log_2 C \in [-20, 20]$, $\log_2 a \in [-20, 20]$ and $b \in [0, 20]$. From those figures we notice that:

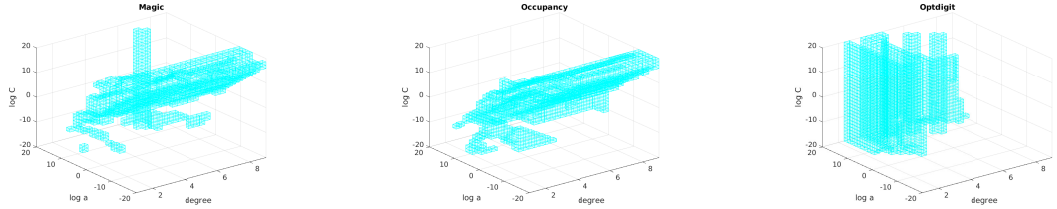


Figure 3.11: 10-percent region for Magic, Occupancy and Optdigit Dataset (Polynomial Kernel with $b = 1$)

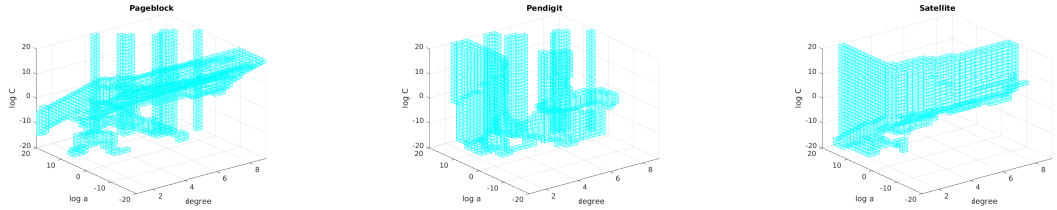


Figure 3.12: 10-percent region for Pageblock, Pendigit and Satellite Dataset (Polynomial Kernel with $b = 1$)

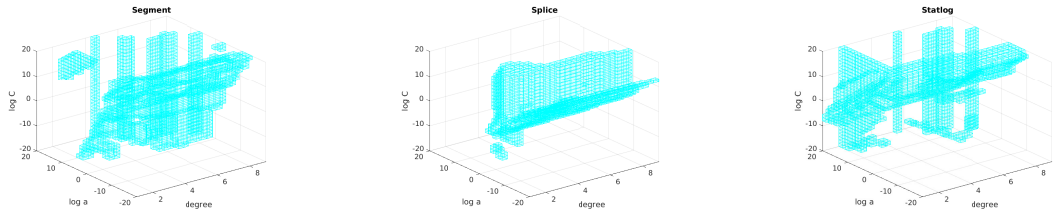


Figure 3.13: 10-percent region for Segment, Splice and Statlog Dataset (Polynomial Kernel with $b = 1$)

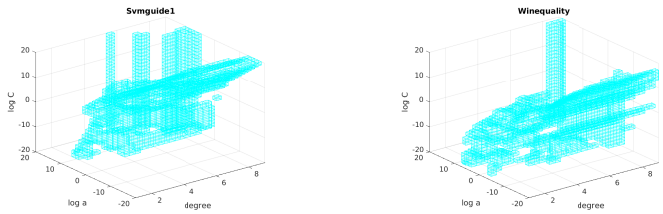


Figure 3.14: 10-percent region for Svmguide1 and Winequality Dataset (Polynomial Kernel with $b = 1$)

- For b, all regions are equally important.

Based on those observation we set the range of the parameters of polynomial kernel to be $\log_2 C \in [-8, 10]$, $\log_2 a \in [-8, 10]$, $b \in [0, 9]$ and $c \in [1, 7]$.

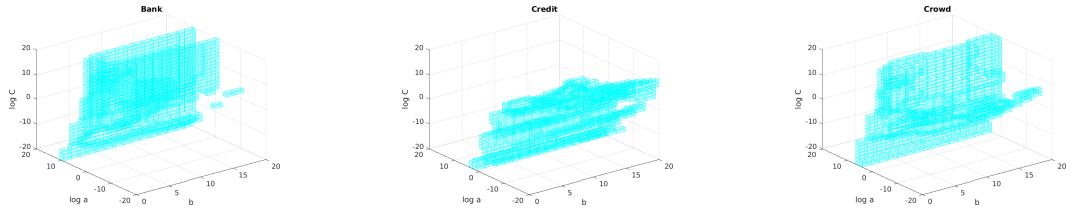


Figure 3.15: 10-percent region for Banknote, Credit and Crowd Dataset (Polynomial Kernel with degree $= 3$)

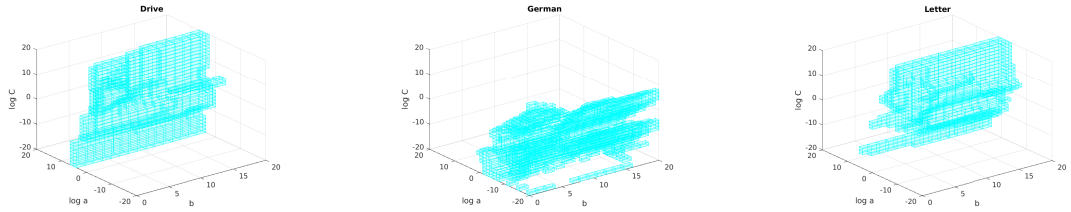


Figure 3.16: 10-percent region for Drive, German and Letter Dataset (Polynomial Kernel with degree $= 3$)

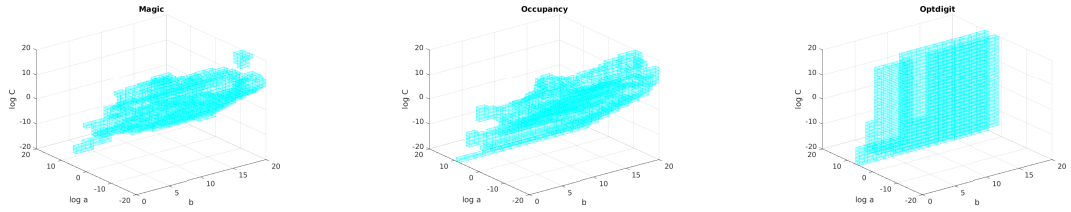


Figure 3.17: 10-percent region for Magic, Occupancy and Optdigit Dataset (Polynomial Kernel with degree $= 3$)

3.4.2 Experimental Design

We use Gaussian kernels, polynomial kernels with $b = 1$ and polynomial kernels in Section 2.3.3 for the two-, three- and four-dimensional experiments, respectively.

Gaussian kernels (2.17) are used for the two-dimensional experiments. Polynomial kernels (2.18) with fixed $b = 1$ are used for the three-dimensional experiments and polynomial kernels are used for the four-dimensional experiments. In each experiment, we search for the optimal specification of the hyper-parameters in the kernels

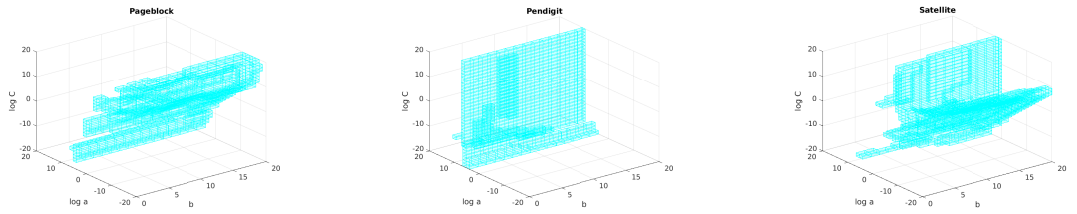


Figure 3.18: 10-percent region for Pageblock, Pendigit and Satellite Dataset (Polynomial Kernel with degree $= 3$)

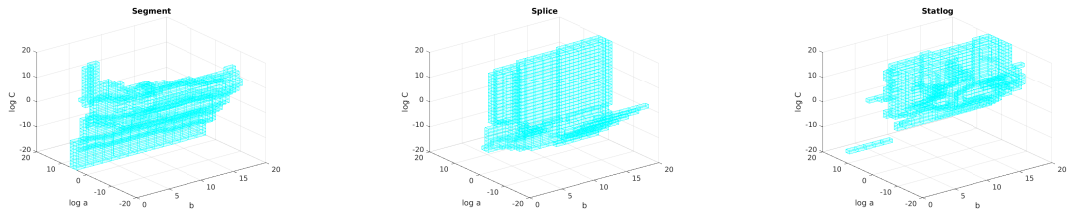


Figure 3.19: 10-percent region for Segment, Splice and Statlog Dataset (Polynomial Kernel with degree $= 3$)

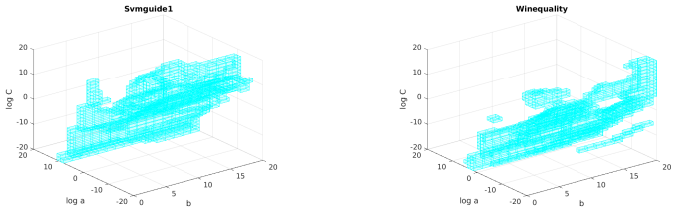


Figure 3.20: 10-percent region for Svmguide1 and Winequality Dataset (Polynomial Kernel with degree $= 3$)

and the regularization parameter C (Section 2.3.1). Table 3.1 is the detailed setting of the hyper-parameters in the kernels and the regularization parameter.

Table 3.1: Setting of hyper-parameters

Kernel	Parameter	Search Region	SEB Grid Size	GS Grid Size	α
$\exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^2}\right)$	$\log\sigma^2$	$[-8,10]$	2	$2/3$	20
Regularization parameter	$\log C$	$[-8,10]$	2	$2/3$	
$(a\langle\mathbf{x}_i,\mathbf{x}_j\rangle+1)^c$	$\log a$	$[-8,10]$	3	1	
	c	$[1,7]$	3	1	15
Regularization parameter	$\log C$	$[-8,10]$	3	1	
$(a\langle\mathbf{x}_i,\mathbf{x}_j\rangle+b)^c$	$\log a$	$[-8,10]$	3	1	
	b	$[0,9]$	3	1	10
	c	$[1,7]$	3	1	
Regularization parameter	$\log C$	$[-8,10]$	3	1	

We compare the prediction accuracy of SEB with the grid search with finer calibration:

1. **SEB** in Section 3.3:
   The ranges are shown in the column "Search Region" and the grid steps for the first sampling are shown in the column "SEB Grid Size" in Table 3.1. We set the α of the α-percent region to 20, 15 and 10 for the two-, three- and four-dimensional experiments, respectively, as shown in the column "α" in Table 3.1.

2. **GS** One-stage grid search with finer sampling points:
   The search ranges are same as SEB and the grid steps are set as in the column "GS Grid Size" in Table 3.1, which are $\frac{1}{3}$ of the grid step of SEB.

For binary classification experiments we use benchmark datasets in Table 1.2. We solve the optimization problem (2.7) using SMO method in Section 2.3.4. For multi-class classification experiments we use benchmark datasets in Table 1.3. We solve the optimization problem (2.11) using the algorithm in (Crammer and Singer [30]). Experiments are repeated 50 times for each dataset.

3.4.3 Results

Tables 3.2 to 3.7 report the results of binary classification carried out on the 17 benchmark datasets for three kernels with the training size $= 100$ and the training size $= 300$ as listed in Table 1.2. Table 3.8 shows the results of multi-class classification carried out on the 10 benchmark datasets for Gaussian kernel as listed in Table 1.3. All the prediction accuracies are on the test datasets. The reported numbers are the average over 50 runs. The column "GS $-$ SEB" shows the results of GS minus SEB on each row. We conduct the paired t-test for each run of experiments to compare the prediction accuracy of SEB to that of GS. If the results of SEB are significantly different from GS (p-values < 0.05), (*) is shown beside the number in the "p-value" column.

Table 3.2: Results of two-dimensional experiments (Gaussian kernel) - training size $= 100$

Dataset	GS	SEB	GS $-$ SEB	p-value	Ratio (SVM runs)
Banknote	99.56 $\pm$ 0.56	99.49 $\pm$ 0.57	0.07 $\pm$ 0.18	$p = 0.014^*$	0.302 (236.5/784)
Credit	78.74 $\pm$ 1.02	78.74 $\pm$ 1.01	-0.00 $\pm$ 0.05	p=0.702	0.429 (336.0/784)
Crowd-sourced	91.69 $\pm$ 1.07	91.68 $\pm$ 1.09	0.02 $\pm$ 0.11	p=0.322	0.303 (237.2/784)
Drive	75.22 $\pm$ 2.35	75.21 $\pm$ 2.36	0.01 $\pm$ 0.08	p=0.322	0.293 (229.5/784)
German	71.87 $\pm$ 2.00	71.68 $\pm$ 1.99	0.19 $\pm$ 0.69	p=0.059	0.301 (235.7/784)
Letter	71.98 $\pm$ 1.97	71.98 $\pm$ 1.97	0.00 $\pm$ 0.01	p=0.322	0.303 (237.4/784)
MAGIC	81.24 $\pm$ 1.41	81.25 $\pm$ 1.42	-0.01 $\pm$ 0.05	p=0.322	0.312 (244.6/784)
Occupancy	98.89 $\pm$ 0.26	98.94 $\pm$ 0.21	-0.04 $\pm$ 0.17	p=0.094	0.321 (252.0/784)
Optdigit	95.29 $\pm$ 1.01	95.29 $\pm$ 1.01	0.00 $\pm$ 0.04	p=0.390	0.27 (212.0/784)
Page Blocks	94.16 $\pm$ 0.90	94.17 $\pm$ 0.91	-0.01 $\pm$ 0.09	p=0.350	0.328 (257.0/784)
Pen Digits	95.25 $\pm$ 1.10	95.24 $\pm$ 1.12	0.01 $\pm$ 0.09	p=0.322	0.277 (217.0/784)
Satellite	93.25 $\pm$ 0.71	93.25 $\pm$ 0.70	-0.01 $\pm$ 0.06	p=0.425	0.3 (235.0/784)
Segment	91.72 $\pm$ 1.46	91.76 $\pm$ 1.47	-0.04 $\pm$ 0.26	p=0.324	0.297 (233.0/784)
Splice	78.80 $\pm$ 1.73	78.80 $\pm$ 1.74	-0.01 $\pm$ 0.04	p=0.322	0.275 (215.9/784)
Statlog	98.44 $\pm$ 0.95	98.42 $\pm$ 0.98	0.02 $\pm$ 0.12	p=0.304	0.312 (244.6/784)
Svmguide1	95.54 $\pm$ 0.68	95.53 $\pm$ 0.68	0.00 $\pm$ 0.03	p=0.253	0.303 (237.6/784)
Wine Quality	79.47 $\pm$ 0.84	79.46 $\pm$ 0.84	0.01 $\pm$ 0.08	p=0.604	0.381 (298.8/784)

3.4.4 Supplemental Experiments for Different α Values and Grid Sizes

In this section we examine the effects of different values of α and different grid sizes for the overall performance of SEB and Grid Search with the $1/3$ step size of SEB. In

Table 3.3: Results of three-dimensional experiments (Polynomial kernel with $b = 1$) - training size $= 100$

Dataset	GS	SEB	GS − SEB	p-value	Ratio (SVM runs)
Banknote	99.62 ± 0.39	99.52 ± 0.41	0.10 ± 0.35	p=0.052	0.265 (670.8/2527)
Credit	78.85 ± 1.09	78.81 ± 1.07	0.04 ± 0.43	p=0.495	0.242 (611.9/2527)
Crowd-sourced	88.88 ± 1.37	88.80 ± 1.37	0.08 ± 0.41	p=0.182	0.274 (693.4/2527)
Drive	68.15 ± 2.71	67.77 ± 2.81	0.38 ± 1.04	$p = 0.013^*$	0.289 (730.9/2527)
German	72.09 ± 2.26	72.03 ± 2.27	0.06 ± 0.84	p=0.592	0.277 (700.1/2527)
Letter	71.68 ± 2.19	71.64 ± 2.09	0.04 ± 0.44	p=0.531	0.275 (695.4/2527)
MAGIC	81.57 ± 1.43	81.57 ± 1.35	-0.01 ± 0.37	p=0.908	0.261 (659.1/2527)
Occupancy	98.59 ± 0.31	98.57 ± 0.35	0.02 ± 0.19	p=0.537	0.291 (736.0/2527)
Optdigit	94.83 ± 1.25	94.86 ± 1.25	-0.03 ± 0.14	p=0.144	0.306 (774.1/2527)
Page Blocks	94.50 ± 0.74	94.49 ± 0.76	0.01 ± 0.19	p=0.845	0.298 (754.2/2527)
Pen Digits	95.19 ± 1.23	95.24 ± 1.21	-0.05 ± 0.28	p=0.181	0.282 (713.0/2527)
Satellite	92.70 ± 0.64	92.74 ± 0.58	-0.05 ± 0.31	p=0.305	0.26 (657.9/2527)
Segment	90.81 ± 1.74	90.86 ± 1.66	-0.05 ± 0.42	p=0.442	0.288 (727.5/2527)
Splice	78.82 ± 1.53	78.71 ± 1.46	0.11 ± 0.49	p=0.120	0.219 (553.0/2527)
Statlog	98.00 ± 1.25	97.99 ± 1.26	0.02 ± 0.17	p=0.454	0.275 (695.5/2527)
Svmguide1	95.63 ± 0.52	95.68 ± 0.54	-0.05 ± 0.25	p=0.184	0.283 (715.6/2527)
Wine Quality	79.58 ± 0.87	79.64 ± 0.74	-0.06 ± 0.54	p=0.459	0.261 (658.7/2527)

Table 3.4: Results of four-dimensional experiments (Polynomial kernel) - training size $= 100$

Dataset	GS	SEB	GS $-$ SEB	p-value	Ratio (SVM runs)
Banknote	99.66 $\pm$ 0.42	99.60 $\pm$ 0.42	0.06 $\pm$ 0.31	p=0.187	0.189 (4772.5/25270)
Credit	80.35 $\pm$ 1.42	80.34 $\pm$ 1.37	0.01 $\pm$ 0.20	p=0.794	0.171 (4308.9/25270)
Crowd-sourced	88.88 $\pm$ 1.37	88.59 $\pm$ 1.41	0.28 $\pm$ 0.60	$p = 0.001^*$	0.195 (4938.6/25270)
Drive	67.87 $\pm$ 2.34	67.53 $\pm$ 2.28	0.34 $\pm$ 0.86	$p = 0.013^*$	0.202 (5102.9/25270)
German	71.98 $\pm$ 2.12	72.10 $\pm$ 2.01	-0.12 $\pm$ 1.29	p=0.579	0.204 (5160.6/25270)
Letter	71.90 $\pm$ 1.89	71.62 $\pm$ 1.88	0.28 $\pm$ 0.61	$p = 0.002^*$	0.196 (4959.2/25270)
MAGIC	81.83 $\pm$ 1.09	81.80 $\pm$ 1.10	0.03 $\pm$ 0.35	p=0.539	0.192 (4844.5/25270)
Occupancy	98.57 $\pm$ 0.36	98.62 $\pm$ 0.31	-0.05 $\pm$ 0.23	p=0.124	0.213 (5378.9/25270)
Optdigit	94.90 $\pm$ 1.31	94.98 $\pm$ 1.31	-0.08 $\pm$ 0.25	$p = 0.026^*$	0.212 (5357.3/25270)
Page Blocks	94.30 $\pm$ 0.94	94.30 $\pm$ 0.96	-0.00 $\pm$ 0.15	p=0.902	0.213 (5384.1/25270)
Pen Digits	95.31 $\pm$ 1.25	95.18 $\pm$ 1.35	0.13 $\pm$ 0.35	$p = 0.011^*$	0.199 (5041.0/25270)
Satellite	92.68 $\pm$ 0.76	92.56 $\pm$ 0.80	0.13 $\pm$ 0.48	p=0.065	0.196 (4940.7/25270)
Segment	91.23 $\pm$ 1.39	91.14 $\pm$ 1.41	0.09 $\pm$ 0.46	p=0.185	0.211 (5341.4/25270)
Splice	78.77 $\pm$ 1.63	78.34 $\pm$ 1.86	0.42 $\pm$ 0.88	$p = 0.001^*$	0.162 (4081.3/25270)
Statlog	98.10 $\pm$ 1.09	98.06 $\pm$ 1.09	0.04 $\pm$ 0.30	p=0.412	0.2 (5050.1/25270)
Svmguide1	95.65 $\pm$ 0.57	95.64 $\pm$ 0.55	0.01 $\pm$ 0.15	p=0.784	0.211 (5333.8/25270)
Wine Quality	79.29 $\pm$ 0.82	79.35 $\pm$ 0.71	-0.06 $\pm$ 0.53	p=0.400	0.182 (4602.4/25270)

Table 3.5: Results of two-dimensional experiments (Gaussian kernel) - training size $= 300$

Dataset	GS	SEB	GS $-$ SEB	p-value	Ratio (SVM runs)
Banknote	99.83 $\pm$ 0.27	99.87 $\pm$ 0.27	-0.04 $\pm$ 0.28	p=0.298	0.315 (247.3/784)
Credit	79.21 $\pm$ 1.12	79.21 $\pm$ 1.11	0.00 $\pm$ 0.03	p=0.322	0.396 (310.7/784)
Crowd-sourced	93.87 $\pm$ 0.63	93.85 $\pm$ 0.63	0.02 $\pm$ 0.14	p=0.323	0.298 (233.5/784)
Drive	87.71 $\pm$ 1.38	87.69 $\pm$ 1.40	0.02 $\pm$ 0.14	p=0.322	0.28 (219.8/784)
German	74.39 $\pm$ 2.57	74.29 $\pm$ 2.65	0.10 $\pm$ 0.71	p=0.322	0.337 (264.1/784)
Letter	81.23 $\pm$ 1.30	81.24 $\pm$ 1.35	-0.01 $\pm$ 0.10	p=0.322	0.285 (223.6/784)
MAGIC	83.53 $\pm$ 0.88	83.54 $\pm$ 0.89	-0.01 $\pm$ 0.07	p=0.322	0.312 (244.8/784)
Occupancy	98.95 $\pm$ 0.23	98.96 $\pm$ 0.23	-0.01 $\pm$ 0.12	p=0.549	0.338 (265.4/784)
Optdigit	97.88 $\pm$ 0.47	97.88 $\pm$ 0.47	-0.00 $\pm$ 0.03	p=0.253	0.27 (211.3/784)
Page Blocks	95.23 $\pm$ 0.58	95.22 $\pm$ 0.57	0.00 $\pm$ 0.02	p=0.322	0.326 (255.5/784)
Pen Digits	97.84 $\pm$ 0.42	97.84 $\pm$ 0.41	-0.00 $\pm$ 0.04	p=0.563	0.284 (222.5/784)
Satellite	94.08 $\pm$ 0.56	94.07 $\pm$ 0.57	0.01 $\pm$ 0.05	p=0.148	0.306 (240.0/784)
Segment	95.18 $\pm$ 0.86	95.15 $\pm$ 0.89	0.02 $\pm$ 0.14	p=0.243	0.294 (230.8/784)
Splice	84.86 $\pm$ 0.92	84.89 $\pm$ 0.95	-0.03 $\pm$ 0.16	p=0.213	0.271 (212.3/784)
Statlog	99.54 $\pm$ 0.35	99.54 $\pm$ 0.35	0.00 $\pm$ 0.04	p=0.825	0.314 (246.2/784)
Svmguide1	95.94 $\pm$ 0.48	95.92 $\pm$ 0.51	0.01 $\pm$ 0.08	p=0.322	0.308 (241.3/784)
Wine Quality	80.04 $\pm$ 0.92	80.12 $\pm$ 0.81	-0.07 $\pm$ 0.31	p=0.104	0.353 (276.9/784)

Table 3.6: Results of three-dimensional experiments (Polynomial kernel with $b = 1$) - training size $= 300$

Dataset	GS	SEB	GS − SEB	p-value	Ratio (SVM runs)
Banknote	99.82 ± 0.37	99.82 ± 0.30	0.00 ± 0.47	p=0.962	0.271 (684.7/2527)
Credit	81.28 ± 1.32	81.30 ± 1.37	-0.02 ± 0.37	p=0.697	0.259 (653.8/2527)
Crowd-sourced	92.37 ± 0.79	92.43 ± 0.83	-0.05 ± 0.29	p=0.185	0.277 (700.1/2527)
Drive	73.23 ± 1.43	73.25 ± 1.35	-0.01 ± 0.60	p=0.884	0.287 (724.1/2527)
German	74.53 ± 3.40	74.47 ± 3.38	0.06 ± 0.94	p=0.654	0.293 (739.2/2527)
Letter	79.53 ± 1.39	79.43 ± 1.44	0.10 ± 0.30	$p = 0.027^*$	0.269 (678.9/2527)
MAGIC	83.49 ± 1.04	83.45 ± 1.04	0.03 ± 0.44	p=0.607	0.253 (639.6/2527)
Occupancy	98.93 ± 0.23	98.94 ± 0.21	-0.01 ± 0.14	p=0.678	0.276 (698.3/2527)
Optdigit	97.86 ± 0.58	97.91 ± 0.51	-0.05 ± 0.25	p=0.174	0.311 (784.9/2527)
Page Blocks	95.52 ± 0.63	95.56 ± 0.68	-0.04 ± 0.34	p=0.421	0.29 (731.6/2527)
Pen Digits	97.87 ± 0.45	97.86 ± 0.47	0.01 ± 0.18	p=0.687	0.292 (738.4/2527)
Satellite	93.52 ± 0.51	93.51 ± 0.47	0.00 ± 0.20	p=0.927	0.261 (659.2/2527)
Segment	94.66 ± 0.99	94.68 ± 1.01	-0.03 ± 0.37	p=0.599	0.291 (734.5/2527)
Splice	84.90 ± 0.89	84.83 ± 0.81	0.07 ± 0.52	p=0.339	0.236 (597.6/2527)
Statlog	99.27 ± 0.40	99.22 ± 0.37	0.04 ± 0.15	$p = 0.043^*$	0.269 (679.8/2527)
Svmguide1	96.02 ± 0.47	96.12 ± 0.40	-0.10 ± 0.29	$p = 0.016^*$	0.267 (674.3/2527)
Wine Quality	79.91 ± 0.94	79.91 ± 0.88	0.00 ± 0.44	p=0.983	0.269 (680.8/2527)

Table 3.7: Results of four-dimensional experiments (Polynomial kernel) - training size $= 300$

Dataset	GS	SEB	GS − SEB	p-value	Ratio (SVM runs)
Banknote	99.92 ± 0.19	99.89 ± 0.20	0.03 ± 0.18	p=0.269	0.202 (5116.9/25270)
Credit	81.09 ± 1.42	81.01 ± 1.53	0.07 ± 0.34	p=0.143	0.197 (4987.4/25270)
Crowd-sourced	92.54 ± 0.78	92.36 ± 0.83	0.19 ± 0.40	$p = 0.002^*$	0.177 (4463.3/25270)
Drive	72.26 ± 2.03	72.30 ± 2.07	-0.03 ± 0.42	p=0.706	0.195 (4937.0/25270)
German	74.19 ± 2.83	74.29 ± 2.54	-0.10 ± 1.50	p=0.640	0.199 (5039.0/25270)
Letter	79.15 ± 1.40	79.18 ± 1.46	-0.02 ± 0.57	p=0.771	0.204 (5157.6/25270)
MAGIC	83.56 ± 0.93	83.52 ± 0.95	0.04 ± 0.15	p=0.143	0.193 (4886.6/25270)
Occupancy	98.75 ± 0.19	98.74 ± 0.22	0.01 ± 0.12	p=0.617	0.214 (5414.9/25270)
Optdigit	97.84 ± 0.56	97.83 ± 0.56	0.01 ± 0.20	p=0.653	0.21 (5296.8/25270)
Page Blocks	95.69 ± 0.52	95.58 ± 0.61	0.11 ± 0.31	$p = 0.012^*$	0.21 (5299.6/25270)
Pen Digits	97.73 ± 0.50	97.76 ± 0.48	-0.03 ± 0.19	p=0.238	0.206 (5202.5/25270)
Satellite	93.47 ± 0.53	93.52 ± 0.55	-0.05 ± 0.32	p=0.266	0.201 (5079.5/25270)
Segment	94.27 ± 0.99	94.35 ± 1.02	-0.08 ± 0.31	p=0.077	0.213 (5382.1/25270)
Splice	84.79 ± 1.31	84.49 ± 1.13	0.30 ± 0.61	$p = 0.006^*$	0.188 (4747.9/25270)
Statlog	99.44 ± 0.45	99.46 ± 0.48	-0.01 ± 0.17	p=0.565	0.19 (4794.3/25270)
Svmguide1	96.02 ± 0.42	96.02 ± 0.47	-0.00 ± 0.23	p=0.936	0.203 (5139.8/25270)
Wine Quality	80.34 ± 0.70	80.28 ± 0.83	0.06 ± 0.42	p=0.410	0.197 (4974.9/25270)

Table 3.8: Results of two-dimensional experiments (Gaussian kernel) - Multi-class Classification

Dataset	GS	SEB	GS − SEB	p-value	Ratio (SVM runs)
Crowd-sourced	90.08 ± 0.88	90.05 ± 0.89	0.02 ± 0.08	$p = 0.042^*$	0.327 (256.4/784)
Drive	80.24 ± 2.83	80.60 ± 2.35	-0.35 ± 1.77	p=0.164	0.332 (260.6/784)
Letter	70.53 ± 1.15	70.53 ± 1.26	0.00 ± 0.28	p=0.961	0.333 (261.2/784)
Optdigit	95.15 ± 0.75	95.15 ± 0.75	-0.01 ± 0.23	p=0.854	0.339 (265.7/784)
Page Blocks	95.28 ± 0.43	95.29 ± 0.45	-0.01 ± 0.22	p=0.846	0.34 (266.2/784)
Pen Digits	95.03 ± 1.01	95.01 ± 1.00	0.01 ± 0.07	p=0.190	0.322 (252.4/784)
Satellite	86.41 ± 0.77	86.43 ± 0.78	-0.02 ± 0.13	p=0.297	0.331 (259.6/784)
Segment	93.84 ± 0.90	93.84 ± 0.94	0.00 ± 0.36	p=0.972	0.328 (257.2/784)
Statlog	99.58 ± 0.26	99.59 ± 0.25	-0.02 ± 0.09	p=0.296	0.317 (248.9/784)
Wine Quality	55.17 ± 1.30	55.12 ± 1.40	0.05 ± 0.37	p=0.415	0.331 (259.4/784)

general when grid sizes approach zero, the difference of the prediction accuracies between SEB and GS will also approach to zero, since a very small difference between the hyper-parameters such as $\log C = 1.01$ and $\log C = 1.02$ would not generate any difference on the prediction. Similarly, if the value of α approaches to 100%, the difference of the prediction accuracies between SEB and GS will be close to zero, since SEB will examine all points in the Grid Search with the $1/3$ step size of SEB. Therefore the purpose of the experiments in this section is to examine the general behavior toward the extreme cases (grid sizes $\Rightarrow$ 0 or $\alpha \Rightarrow 100\%$) and the difference of the behavior among datasets.

The Effects of Different Values of α

We examine the effects of different α values for Gaussian kernel, for Polynomial kernel with $b = 1$ and for Polynomial kernel using three datasets; Occupancy, Optdigit and Statlog randomly chosen from the 17 benchmark datasets. Figure 3.21 shows the prediction accuracy of SEB and GS of different values of α for Gaussian kernel. We set the sizes of α to $\{10, 20, 30, 50, 70, 90\}$. Figure 3.22 shows the ratio of the number of SVM runs for SEB and for GS for different values of α. The value of α is in direct proportion to the ratio, for instance the ratio is about 0.3 for the 30-percent region. Figure 3.23 and Figure 3.24 show the prediction accuracy of SEB and GS of different values of α for Polynomial kernel with $b = 1$ and for Polynomial kernel.

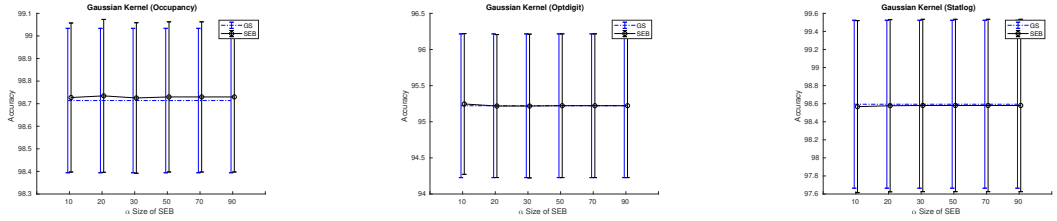


Figure 3.21: The effect of different α sizes (Gaussian Kernel)

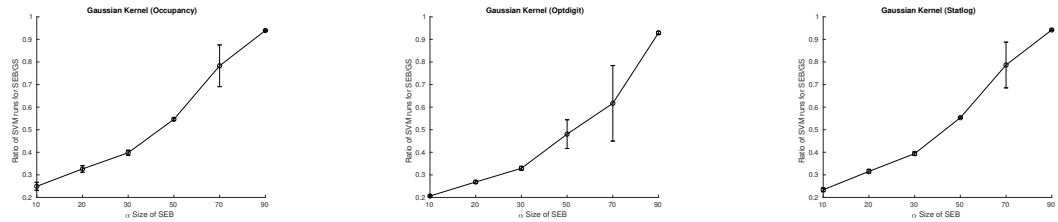


Figure 3.22: The ratio of SVM runs for SEB and GS for different α sizes (Gaussian Kernel)

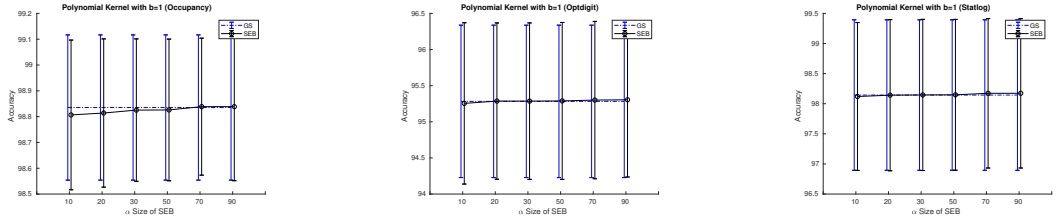


Figure 3.23: The effect of different α sizes (Polynomial Kernel with $b = 1$)

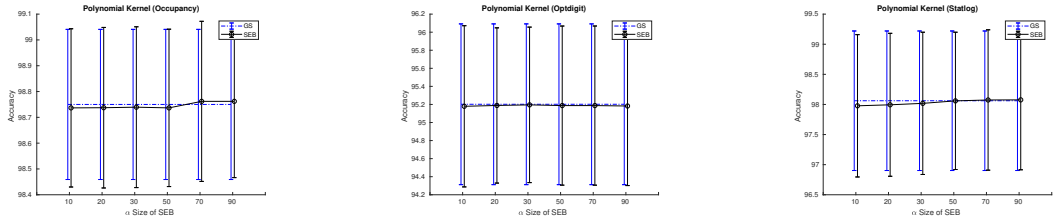


Figure 3.24: The effect of different α sizes (Polynomial Kernel)

The Effects of Different Grid Sizes

We examine the effects of different grid sizes for Gaussian kernel, Polynomial kernel with $b = 1$ and Polynomial kernel, using Bank and Statlog dataset. We compare the accuracy of SEB with GS whose grid step is $1/3$ of SEB which we denote "GS" as above. As a reference, we also show the accuracy of GS with the same step size of SEB, which is denoted as "GS2" in this experiment.

For c (degree) of Polynomial kernel, we set the step size $= 3$ for SEB and GS2 and the step size $= 1$ for GS. For other hyper-parameters, we set the step size of SEB and GS2 as follows:

- $\{0.3, 0.6, 0.9, 3, 6\}$ for Gaussian kernel and Polynomial kernel with $b = 1$
- $\{0.9, 3, 6\}$ for Gaussian kernel and Polynomial kernel.

(The step size of GS is three times smaller than those values.) Experiments are repeated 50 times for each dataset.

Figure 3.25 shows the prediction accuracy of SEB, GS and GS2 for different grid sizes for Gaussian kernel and Figure 3.26 shows the number of SVM runs for those three methods. Figure 3.27 - Figure 3.30 show the results for Polynomial kernel with $b = 1$ and Polynomial kernel.

3.4.5 Discussion

We conducted two-, three- and four-dimensional experiments using two types of kernels; Gaussian kernels, Polynomial kernels with $b = 1$, and Polynomial kernels and

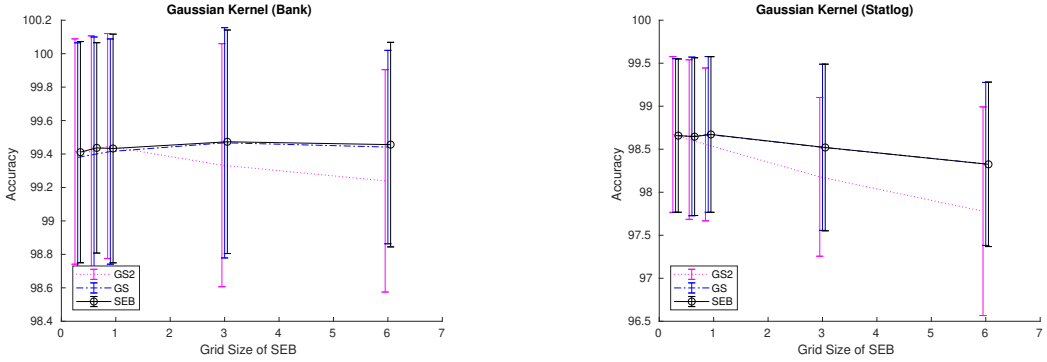


Figure 3.25: The effect of different grid sizes (Gaussian Kernel, $\alpha = 20$)

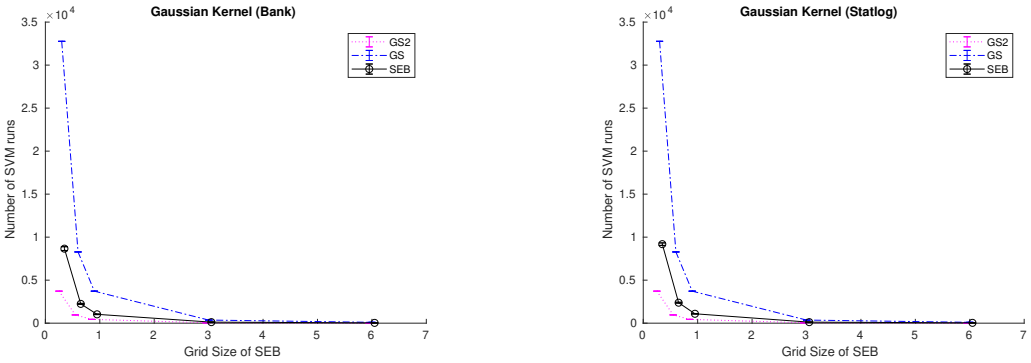


Figure 3.26: Number of SVM runs for different grid sizes (Gaussian Kernel, $\alpha = 20$)

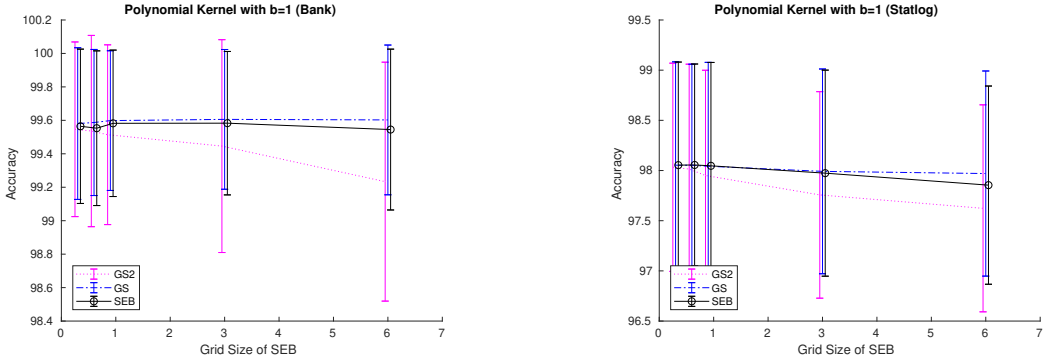


Figure 3.27: The effect of different grid sizes (Polynomial Kernel with $b = 1$, $\alpha = 15$)

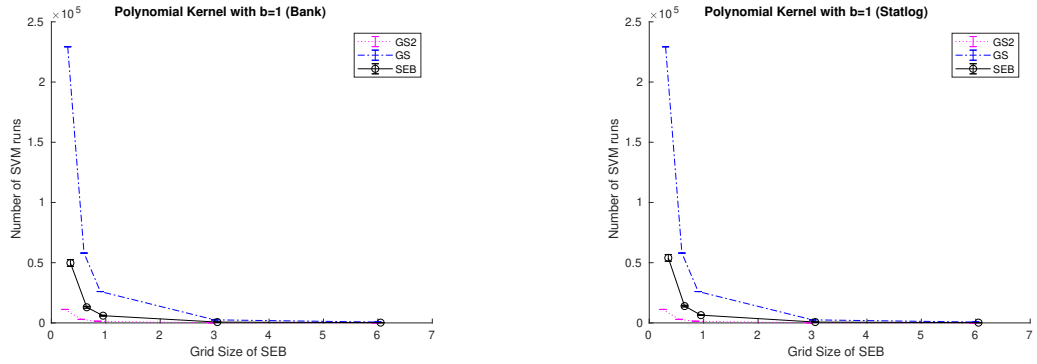


Figure 3.28: Number of SVM runs for different grid sizes (Polynomial Kernel with $b = 1$, $\alpha = 15$)

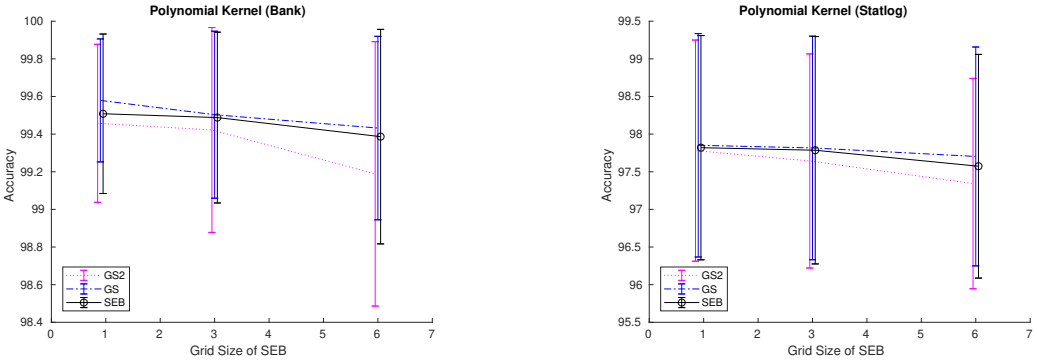


Figure 3.29: The effect of different grid sizes (Polynomial Kernel, $\alpha = 10$)

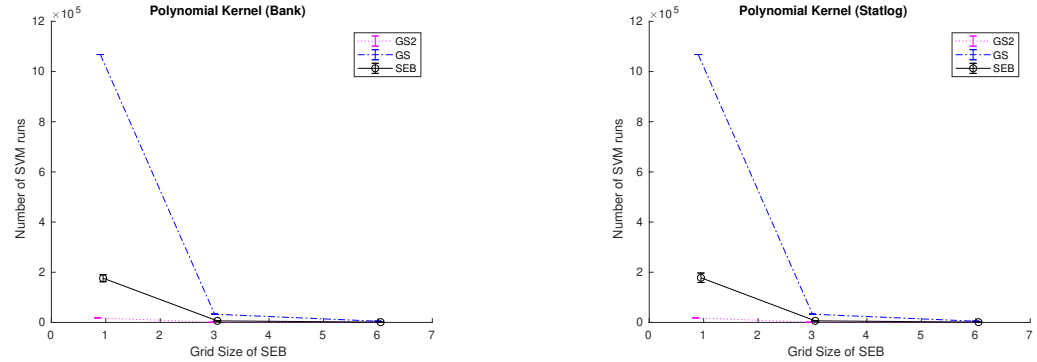


Figure 3.30: Number of SVM runs for different grid sizes (Polynomial Kernel, $\alpha = 10$)

different training sizes $= 100$ and $= 300$. We reported the results in Table 3.2 to Table 3.7 in Section 3.4.3. We also examined the effects of different grid sizes and α values.

In Table 3.4 in four-dimensional experiments with training size $= 100$, the results of SEB are significantly better than GS for one dataset and significantly worse than GS for five datasets out of 17 datasets. In Table 3.6 in three-dimensional experiments with training size $= 300$, the results of SEB are significantly better than GS for one dataset and significantly worse than GS for two datasets. In Table 3.7 in four-dimensional experiments with training size $= 300$, the results of SEB are significantly worse than GS for three datasets. All tables, however, indicate that the overall performance of SEB is very close to that of GS with $1/3$ finer grid sizes.

With respect to efficiency, SEB achieved the equivalent prediction accuracy as GS with the saving of 60% to 80% of computation. We achieved the saving by adjusting the size of α which is proportional to the ratio of the SVM runs of SEB to the SVM runs of GS.

In the experiments of the different values of α, Figure 3.21 - Figure 3.24 indicate that the quality of the solution found by the algorithm is robust against the values of α. Note that the higher the value of α, the larger the α-percent region and therefore more points must be evaluated in the second stage of the algorithm. Our experiments show that low values of α (for example 10%) does not deteriorate the quality of the solutions while a lot of saving is achieved by not evaluating unnecessary regions of the space. The size of α is in direct proportion to the ratio of the number of SVM runs of SEB to that of GS (Figure 3.22).

In the experiments of the different grid sizes, we compare the three methods; SEB, GS whose grid step is $1/3$ of SEB which is denoted as "GS", GS with the same grid size as SEB which is denoted as "GS2". In Figure 3.25 the performance of GS2 deteriorates rapidly, but SEB keep the same level of prediction accuracy as GS. Figure 3.26 shows that the number of SVM runs of SEB is much smaller than GS where the step sizes are small. The other figures also show the same pattern. Therefore, SEB behaves more efficiently than GS without sacrificing accuracy.

Given a number n of evaluation points, the time complexity of constructing Bézier surface is $O(n)$ (2.41). The time complexity of computing the α-percent region is also $O(n)$ (2.42). is because, as mentioned earlier, the volume under a Bézier surface can be obtained via a closed-form formula (no numerical approximation is needed).

As for the actual running time of Bézier curve methods it takes an average of 3.95 seconds (and maximum of 5.90 seconds) for 17 datasets in four dimensional polyno-

mial experiments to construct the Bézier surface and compute the α-percent region. Note that this time does not depend on the number of training examples or number of features as it is performed in the hyper-parameter space. It is not a large overhead and the ratio of the actual running time of SEB to that of GS is almost equal to the ratio of the number of SVM runs of SEB to that of GS.

3.5 Summary

We introduced a surface estimation method to detect the most relevant area in the search region by estimating the prediction accuracy surface over the search space. The proposed method makes the grid search more efficient and automatic.

The results in Table 3.2 and 3.7 show that the optimal solutions by the proposed algorithm are in most cases close to the optimal solutions obtained by more exhaustively searching the space. Therefore we will use this method as a benchmark in the following chapters.

Figure 3.21 - Figure 3.24 indicate that the quality of the solution found by the proposed method is robust against the values of α. Figure 3.25 - Figure 3.30 indicate that SEB behaves more efficiently than GS without sacrificing accuracy.

The proposed surface estimation method, however, still suffers from the problem of exponential increase of evaluation points with the increase of the dimensionality of spaces. We tackle this problem using EC methods in the following chapters.

Contributions and achievements:

- We developed a tool based on the Bézier curve methods to estimate the surface of the prediction accuracy in hyper-parameter spaces.
- We developed an algorithm that can use estimated prediction surfaces in order to efficiently identify good regions in the space and find optimal solutions.
- We experimentally show that the proposed methods improve the efficiency of the grid search without sacrificing accuracy.

Chapter 4

Discretization of Continuous PSO

In this chapter we present a unified framework in which discrete PSO is viewed as a variant of continuous PSO. The particles move around in continuous spaces but are evaluated at discrete points. The positions of particles are mapped into the discrete points using "discretization" functions. In this framework the velocity and the position update formula of continuous PSO are preserved for the discrete PSO. We present basic convergence theorems for the discrete PSO and prove that the stability theorems developed for the continuous PSO are also valid for the discrete version without any changes. In this chapter we introduce two types of discretization functions—one for the binary PSO and one for the more general discrete PSO which are not restricted to binary responses. For the binary PSO, the proposed discretization process leads to the "position as probability" models. In experiments we show that the "position as probability" approach outperforms the current "velocity as probability" approach. For the discrete PSO, we test the proposed models using the "Independent Job Scheduling Problems".

This chapter is a preparation of tools which will be used in the later chapters. Since hyper-parameters take continuous values, PSO is more suitable search method than other EC methods such as GA. Even when the search space is discretized, there is still a *total order* between points along each axis. GA, while generally good on discrete problems, disregards the order. PSO also has an advantage that it has a fewer number of parameters which need to be adjusted. We will use the discrete PSO for the hyper-parameter search to improve the efficiency of continuous PSO in Chapter 5. We will also use the binary PSO for the feature subset selection in Chapter 7.

4.1 Introduction

Particle Swarm Optimization (PSO) is a population-based optimization method which produces computational intelligence through social interaction [70]. Since its invention in 1995, PSO has achieved great success in various fields of applications. During its history, however, most researches have focused on the continuous PSO. In this chapter, we examine the discretization of continuous PSO in a unified way. The basic idea is that by mapping points in continuous space into finite evaluation points in the same domain we can construct the discrete version of PSO which inherit all properties of continuous PSO.

Firstly, we start with the binary PSO. In the literature there are two different approaches for the binary PSO. A binary version of PSO was firstly introduced by Kennedy and Eberhart [71] in 1997. In a standard model of binary PSO, the positions of particles are restricted to the vertices of unit hyper-cubes and the velocities are interpreted as the probability that particles move from the current vertices to the other vertices. We call it the "velocity as probability" approach. The problem of this approach is that the position update formula is completely different from that of continuous PSO. Zhen et al. [181] introduced the probability-based binary PSO in which the positions of particles represent the probability of taking the value of 1 and the binary outcomes are stochastically determined based on the probability. In this model the update rules of the velocity and the positions are both preserved. We call it the "position as probability" approach.

In this chapter we derive several binary PSO models from existing continuous PSO and show that the "position as probability" approach is theoretically supported and also outperforms the "velocity as probability" approach. In the experiments we compare the performance of the proposed methods with the current binary PSO models using benchmark functions and "Knapsack Problems".

An important aspect in the experiments of binary PSO is the coding scheme of bit strings. In order to test the performance of newly proposed PSO we commonly use benchmark functions, which provide challenging multi-modal problems with known optimal values. In the case of binary PSO the real solutions for the benchmark functions must be represented by bit strings. In the literature, the binary representation of integers is often used for the transformation. However, it does not preserve the relative distance between the points in spaces. A better choice for the transformation of real solutions would be Gray code, which is a minimal change ordering of bit strings for integers. We will verify this claim experimentally.

Next, we investigate the discrete PSO, which takes a finite but arbitrary number of values (not restricted to binary responses). We consider models in which particles move in the continuous spaces but evaluated at discrete points in the same domain. We propose one type of discrete PSO which is based on "round" functions. As applications of the proposed method we conduct experiments using Independent Job Scheduling Problems and efficacy improvement of continuous PSO.

Lastly, those models in binary PSO and in discrete PSO are formulated in a unified way by introducing discretization functions. This general approach clarifies the nature of the problem. The velocity and position update rules are the same as the continuous PSO, but the pbest (the personal best position) and the gbest (the swarm best position) are updated through the discretization functions. We look upon the binary and the discrete PSO as variants of continuous PSO. In fact, in most cases it is a simple task to derive the discrete version from the continuous PSO and the theories developed for the continuous PSO are valid for the discrete version of PSO.

Gap or weakness in previous research: Continuous PSO and discrete (binary) PSO are looked upon as different subjects and the theorems developed for continuous PSO can not be applied directly to the discrete version of PSO.

Objectives of Chapter 4: We propose a unified approach to combine the continuous PSO and discrete (binary) PSO.

This chapter is organized as follows. In Section 4.2, we start with a review of theorems and models of continuous and binary PSO. In Section 4.3, we propose general models with discretization functions, prove basic convergence theorems for discrete PSO and derive new discrete (including binary) PSO models from existing continuous PSO. In Section 4.4, we conduct a number of experiments to examine the performance and the properties of the proposed methods. We summarize the chapter in Section 4.5.

4.2 Related Work

In this section we review the stability analysis of continuous PSO and a continuous PSO model and a binary PSO model which are most relevant to our proposed methods.

4.2.1 Stability Analysis

One of the most important aspects of the theory of PSO is to investigate the conditions whereby the solutions generated by the algorithm converge to a point. Usually it aims to determine a condition in terms of ω, c_1, c_2 for SPSO in Section 2.5.2 and ω, c_1, c_2, c_3 for FIPS in Section 2.5.2, for which the expectation and the variance of the particle positions are both convergent. It is called an "order-2" stability [121].

Bonyadi and Michalewicz [15] derived a necessary and sufficient condition for the convergence of expectation of SPSO as:

$$-1 < \omega < 1, \qquad 0 < c_1 + c_2 < 4(\omega + 1). \tag{4.1}$$

They derived (4.1) under the assumption that x^t is updated as

$$x^{t+1} = lx^t - \omega x^{t-1} + \phi_1 p + \phi_2 g \tag{4.2}$$

where $l = \omega + 1 - \phi_1 - \phi_2$ and $\omega, \phi_1, \phi_2, p, g$ are random numbers. (By substituting (2.31) into (2.34), the positions update rule of SPSO has the same form as (4.2).) The fixed point of the expectation of SPSO is computed as $\frac{c_1\mu_p + c_2\mu_g}{c_1+c_2}$, where μ_α is a mean of α. Bonyadi and Michalewicz [15] also derived a necessary condition for the convergence of variance as follows.

$$-1 < \omega < 1, \qquad 0 < c_1 + c_2 < 24\frac{1-\omega^2}{7-5\omega} \tag{4.3}$$

It is only a necessary condition but they experimentally verified that it is also (most likely) a sufficient condition. Common choices of parameters for the standard PSO, such as $\omega = 0.729, c_1 = c_2 = 1.49$ [41], and $\omega = 0.6, c_1 = c_2 = 1.7$ [158], satisfy the condition (4.1) and (4.3).

Cleghorn and Engelbrecht [25] derived a necessary condition for the order-2 stability of FIPS. Assuming that the pbest $\mathbf{p}_k^t$ are not updated during the run, $|T_i^t| = \kappa$ for every i, they computed a necessary condition

$$-1 < \omega < 1, \qquad 0 < c_1 + \ldots + c_\kappa < \frac{12\kappa\,(1-\omega^2)}{3\kappa + 1 + \omega\,(1-3\kappa)}. \tag{4.4}$$

Although this is only a necessary condition, the authors experimentally verified that it is also (almost likely) a sufficient condition [25]. Figure 4.1 shows the boundary curve of (4.23) with respect to $c = c_1 + \ldots + c_\kappa$ and ω for fixed values of κ.

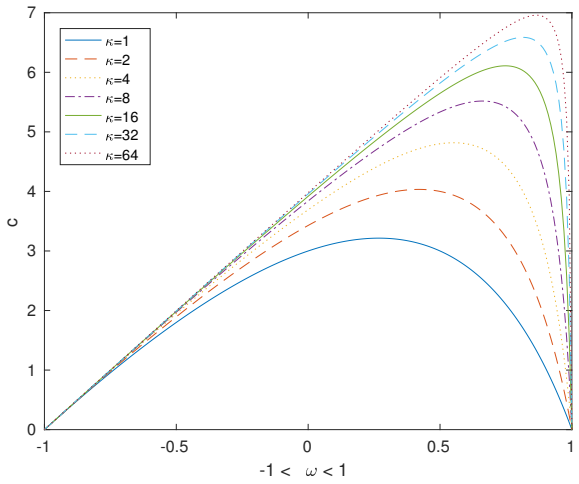


Figure 4.1: The convergence region for $\kappa = 1, 2, 4, 8, 16, 32, 64$

4.2.2 Locally Convergent Rotationally Invariant PSO (LcRiPSO ⇒ SLcRiPSO, FLcRiPSO)

Bonyadi et al. [14] proposed the Locally convergent Rotationally invariant PSO (LcRiPSO), whose velocity update rule is formulated based on the model of FIPS (2.35) using a perturbation function f:

$$\mathbf{v}_i^{t+1} = \omega \mathbf{v}_i^t + \sum_{k \in T_i^t} \frac{c_k r_{k,i}^t}{|T_i^t|} \left(f_{k,i}^t(\mathbf{p}_k^t) - \mathbf{x}_i^t \right) \tag{4.5}$$

where T_i^t is the set of indexes of the neighbors of the particle i and i is assumed to be in T_i^t. (Each particle refers to its own pbest for the velocity update.) Note that for each k, t and i; the same random number $r_{k,i}^t$ is used for all dimensions of the space $j = 1, \ldots, d'$.

This model is formulated to address the common issues of PSO.

- Stagnation: The stagnation occurs when $\mathbf{x}_i^t = \mathbf{p}_i^t = \mathbf{g}_i^t$ and $\mathbf{v}_i^t = \mathbf{0}$ for all particles i at the iteration t. In this case no further improvement can happen.
- Dimensional stagnation: The stagnation takes place in one dimension.
- Small size problem: All particles stop moving in the early stages of the optimization process when the swarm size is small.
- Locally convergence problem: The gbest may not be optimal in its neighborhood.
- Scale problem: The performance of PSO radically deteriorates when the number of dimension d' is large.

- Rotational variance: The performance of PSO deteriorates if the search space is rotated.

Bonyadi et al. [14] proved that if f satisfies the following two conditions, LcRiPSO solves all of the six problems.

1. Locally convergence condition:

$$\forall \mathbf{y} \in S \; \exists A_{\mathbf{y}} \; \forall \mathbf{z} \in A_{\mathbf{y}} \; \forall \gamma > 0 \quad P\left(\|f(\mathbf{y}) - \mathbf{z}\| < \gamma\right) > 0 \tag{4.6}$$

   where $A_{\mathbf{y}}$ is an open set which contains $\mathbf{y}$. This condition means that for any input $\mathbf{y}$ in the search space S, there exists a neighborhood $A_{\mathbf{y}}$ such that $f(\mathbf{y})$ can be at any location within $A_{\mathbf{y}}$ with non-zero probability.

2. Rotational invariance condition:

$$\forall \mathbf{y} \in \mathbb{R}^d \quad sQf(\mathbf{y}) + \mathbf{b} = f\left(sQ\mathbf{y} + \mathbf{b}\right) \tag{4.7}$$

   where Q is a rotation matrix (an orthonormal matrix with determinant 1), s is a scaler (> 0) and $\mathbf{b}$ is a translation vector.

As a specific example of f, Bonyadi et al. [14] proposed

$$f(\mathbf{p}_i^t) = N\left(\mathbf{p}_i^t, \sigma^2 I\right) \tag{4.8}$$

where I is an identity matrix and $N\left(\mathbf{p}_i^t, \sigma^2 I\right)$ is a random vector generated from the normal distribution with the mean vector $\mathbf{p}_i^t$ and the covariance matrix $\sigma^2 I$. We denote the specific model of LcRiPSO with the "gbest" topology of the standard PSO (SPSO) as SLcRiPSO and the model with the ring topology of FIPS as FLcRiPSO.

4.2.3 Probability-based Binary PSO

A binary PSO which represents the "position as probability" approach was firstly proposed by Zhen et al. [181]. The positions (which represent probabilities) of particles are initialized as

$$\dot{\mathbf{x}}_i^0 = \left(\dot{x}_{i,1}^0, \ldots, \dot{x}_{i,d'}^0\right) = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right) \quad .$$

The velocities and positions $\dot{\mathbf{x}}$ are updated according to (2.30) and (2.31) respectively. The actual outcome of a particle is obtained stochastically as:

$$x_{i,j}^t = \begin{cases} 1, & \text{if } r < \dot{x}_{i,j}^t \\ 0, & \text{otherwise.} \end{cases}$$

where r is a random number generated uniformly in $[0, 1]$. The pbest and the gbest vectors are updated according to (2.32) and (2.33) respectively. We denote this method as "PBPSO" in the experiments in Section 4.4.1.

In order to avoid premature convergence, the authors also propose a mutation operator that flips the bits in the outcome with a random probability P_m as follows:

$$x_{i,j}^{t+1} = \begin{cases} \bar{x}_{i,j}^{t}, & \text{if } r < P_m \\ x_{i,j}^{t}, & \text{otherwise .} \end{cases} \tag{4.9}$$

Some authors also proposed probability-based binary PSOs. Strasser et al. [149] proposed a nested model in which each particle position represents the probability distribution of the states of a variable. Wang et al. [181], [180] proposed a model in which the pbest and the gbest take any values in the space. It looks similar but it is quite different from our method in which both pbest and gbest can take only a finite number of values. In their model, since the evaluation of pbest and gbest is stochastic, many points are evaluated with the same fitness values and the same points are evaluated differently from time to time. It results in a slow convergence and a fluctuation behavior. We will check it in Section 4.4.1. We denote this binary PSO model as "WBPSO" in the experiments.

4.2.4 Global convergence of PSO

Bergh and Engelbrecht [159] introduced two methods to make a locally convergent PSO to be also globally convergent. The first one is a "random particle approach" in which randomized particles are added to the swarm. Particles are randomized by "resetting its position to a random position in search space periodically". [159] Another method is a "multi-start approach", in which all particles are initialized to random positions after the swarm has locally converged to a point. The convergence to a local minimum is detected by computing the ratio of the radius of the swarm to the radius of the whole search space; by counting the number of particles within the sphere with some threshold radius centered on the gbest; or by computing the rate of change of the objective function.

4.3 Proposed Methods

In this section we formulate a general model of discrete (including binary) PSO, present convergence theorems in discrete spaces and derive some models of discrete PSOs

from the existing continuous PSOs.

4.3.1 General model

In Section 2.5, we stated a model of continuous PSO as follows.

$$\text{minimize:} \quad F(\mathbf{x}), \quad \mathbf{x} = (x_1, \ldots, x_{d'}) \in S \subseteq \mathbb{R}^{d'} \tag{4.10}$$

where the search space S is a hyper-cube in $\mathbb{R}^{d'}$,

$$l_j \leq x_j \leq u_j, \qquad l_j, u_j \in \mathbb{R}, \quad j = 1, \ldots, d'.$$

We formulate our discrete PSO using the same model. The velocity is updated as (2.34) and the positions of particle are updated as (2.31). However, the fitness value of a particle i at iteration t is computed as $F(\psi(\mathbf{x}_i^t))$ using a discretization function ψ which mapps a position vector $\mathbf{x}$ in the continuous space into a vector $\psi(\mathbf{x})$ in a discrete space. For instance, let's consider the feature selection problems using binary PSO. 1 and 0 in the position of a particle represent "presence" and "absence" of the features. $F()$ is a fitness evaluation function for the set of features. Suppose that $\mathbf{x}$ represents an underlying probability and the binary response (1 or 0) are determined stochastically by the probability. In this case ψ maps the underlying probability $\mathbf{x}$ to the binary response. Two types of discretization function ψ will be defined below. Instead of (2.32), the pbest of the particle i is updated as

$$\mathbf{p}_i^{t+1} = \begin{cases} \psi(\mathbf{x}_i^{t+1}) & \text{if } F(\psi(\mathbf{x}_i^{t+1})) < F(\mathbf{p}_i^t) \\ \mathbf{p}_i^t & \text{otherwise .} \end{cases} \tag{4.11}$$

In the following sections we construct the discrete PSO from SPSO and LcRiPSO. To connect the discrete PSO to the continuous PSO, we use discretization functions which map points in a continuous space to discrete points.

Particles move within a hyper-cube S in (4.10) and $l_j \leq x_j \leq u_j$ in dimension j. Suppose that there are n_j points $\mathbf{a}_j = \left(a_{j,1}, \ldots, a_{j,n_j}\right)$ in the interval $[l_j, u_j]$ such that

$$l_j \leq a_{j,1} < \ldots < a_{j,n_j} \leq u_j. \tag{4.12}$$

In binary cases, we set $n_j = 2$ and $a_{j,1} = 0, a_{j,2} = 1$ for $j = 1, \ldots, d'$. We call those points "evaluation points". Let $\mathbf{D}$ be the Cartesian products of $\mathbf{a}_j$,

$$\mathbf{D} = \mathbf{a}_1 \times \ldots \times \mathbf{a}_{d'} \tag{4.13}$$

where $\mathbf{a}_j \times \mathbf{a}_k = \{(a, b) \mid a \in \mathbf{a}_j, b \in \mathbf{a}_k\}$. We also call vectors $\mathbf{a} = (a_1, \ldots, a_{d'})$ in $\mathbf{D}$ "evaluation points", although it is slightly an abuse of notation.

For $\mathbf{x} \in \mathbb{R}$, discretization functions $\psi_j(x_j), j = 1, \ldots, d'$ map an element x_j in the dimension j to an evaluation point in $[l_j, u_j]$. For $\mathbf{x} \in \mathbb{R}^{d'}$, we define

$$\psi(\mathbf{x}) = (\psi_1(x_1), \ldots, \psi_{d'}(x_{d'})) \tag{4.14}$$

In this chapter, we propose two types of discretization functions. The first one is a "stochastic" function. Suppose that $x \in [a_{j,k-1}, a_{j,k}]$.

$$\psi_j(x) = \begin{cases} a_{j,k} & \text{if } r <= \frac{x - a_{j,k-1}}{a_{j,k} - a_{j,k-1}} \\ a_{j,k-1} & \text{if } r > \frac{x - a_{j,k-1}}{a_{j,k} - a_{j,k-1}} \end{cases} \tag{4.15}$$

where r is a random number generated uniformly in $[0, 1]$. The operation $\frac{x - a_{j,k-1}}{a_{j,k} - a_{j,k-1}}$ maps the interval $[a_{j,k-1}, a_{j,k}]$ to $[0, 1]$ and the random process $\psi_j(x)$ takes the value $a_{j,k}$ with the probability $\frac{x - a_{j,k-1}}{a_{j,k} - a_{j,k-1}}$ and takes the value $a_{j,k-1}$ with the probability $1 - \frac{x - a_{j,k-1}}{a_{j,k} - a_{j,k-1}}$. If $x < a_{j,1}$, $\psi_j(x) = a_{j,1}$ and if $x > a_{j,n_j}$, $\psi_j(x) = a_{j,n_j}$. We use this function for binary PSO.

The second one is a "round" function. Suppose that $x \in [a_{j,k-1}, a_{j,k}]$.

$$\psi_j(x) = \begin{cases} a_{j,k} & \text{if } x \in [\frac{a_{j,k-1} + a_{j,k}}{2}, a_{j,k}] \\ a_{j,k-1} & \text{if } x \in [a_{j,k-1}, \frac{a_{j,k-1} + a_{j,k}}{2})] \end{cases} \tag{4.16}$$

$\psi_j(x)$ takes the value $a_{j,k}$ if $x \in [\frac{a_{j,k-1} + a_{j,k}}{2}, a_{j,k}]$ and takes the value $a_{j,k-1}$ if $x \in [a_{j,k-1}, \frac{a_{j,k-1} + a_{j,k}}{2})]$. If $x < a_{j,1}$, $\psi_j(x) = a_{j,1}$ and if $x > a_{j,n_j}$, $\psi_j(x) = a_{j,n_j}$. We use this function for discrete PSO.

The idea of discretization function (4.15) for binary PSO is to estimate the 0-1 response generated by a random process with some underlying probability distribution. It is used in PBPSO in Section 4.3.1 and also a common idea in statistics such as logistic regression. The idea of discretization function (4.16) for discrete PSO is to directly apply the continuous PSO to discrete problems by rounding off the values. It is also used in integer programming problems (Laskari et al. [87], Sahoo et al. [133]).

4.3.2 Global and Local Convergence Theorem

For a set of evaluation points $\mathbf{D}$ in (4.13) and a point $\mathbf{c} \in \mathbf{D}$, if

$$\forall \mathbf{x} \in \mathbf{D} \qquad F(\mathbf{c}) \leq F(\mathbf{x}),$$

then $\mathbf{c}$ is a global minimizer $\mathbf{o}_G$ in $\mathbf{D}$.

Let $n_\delta(\mathbf{y})$ be the set of all points in a closed sphere with the center $\mathbf{y}$ and the radius δ. For $\mathbf{c} \in \mathbf{D}$, let $\mathbf{D}_\delta(\mathbf{c}) = \mathbf{D} \cap n_\delta(\mathbf{c})$. If

$$\exists \delta \ \ \forall \mathbf{x} \in \mathbf{D}_\delta(\mathbf{c}) \qquad F(\mathbf{c}) \leq F(\mathbf{x}),$$

then $\mathbf{c}$ is a local minimizer $\mathbf{o}_L$ in $\mathbf{D}$.

Since $\mathbf{D}$ is a discrete space, if δ is too small, $\mathbf{D}_\delta(\mathbf{c})$ may contain only $\mathbf{c}$. Therefore in binary case we use $\delta \geq 1$. In general we set $\delta \geq \delta^*$ and δ^* is computed as the maximum distance of adjacent points on each axis (except for the boundary points). Using the notation in (4.12),

$$\delta_{j,k} = \|a_{j,k} - a_{j,k-1}\|$$
$$\delta^* = \max_j \max_k \delta_{j,k}$$

$\mathbf{D}_{\delta^*}(\mathbf{c})$ includes both sides of adjacent evaluation points along each axis.

We define a general form of a stochastic algorithm (GSA) based on ([144], [14]). Algorithm GSA in $\mathbf{D}$:

1. Initialize $\mathbf{p}^0$ from the search space $\mathbf{D}$ and set $t = 1$.
2. Generate a random sample $\mathbf{z}^t$ from $\mathbf{D}$.
3. Using a map $h : \mathbf{D} \times \mathbf{D} \Rightarrow \mathbf{D}$, generate the candidate solution $\mathbf{p}^t = h(\mathbf{p}^{t-1}, \mathbf{z}^t)$, set $t = t + 1$ and go to 2.

GSA and the map h may satisfy the following conditions.

1. Condition C1:
   GSA satisfies the condition C1 if
   $$\mathbf{p}^t = h(\mathbf{p}^{t-1}, \mathbf{z}^t) = \begin{cases} \mathbf{z}^t & \text{if } F(\mathbf{z}^t) < F(\mathbf{p}^{t-1}) \\ \mathbf{p}^{t-1} & \text{otherwise} \end{cases}$$
2. Condition C2:
   Let $\mathcal{I}$ be a subset of natural numbers $\mathbb{N}^+$ with infinite elements. GSA satisfies the condition C2 if
   $$\exists \mathcal{I} \subset \mathbb{N}^+ \ \ \forall t \in \mathcal{I} \ \ \forall \mathbf{c} \in \mathbf{D} \qquad P\left(\mathbf{z}^t = \mathbf{c}\right) > 0$$
   (For infinitely many $t > 0$ and for any $\mathbf{c} \in \mathbf{D}$, the probability of $\mathbf{z}^t$ = $\mathbf{c}$ is not zero.)

3. Condition C3:
   GSA satisfies the condition C3 if

$$\forall t \geq 0 \;\; \exists \delta \geq \delta^* \;\; \forall \mathbf{c} \in \mathbf{D}_\delta(\mathbf{z}^t) \qquad P\left(\mathbf{z}^{t+1} = \mathbf{c}\right) > 0$$

   (For any $\mathbf{c} \in \mathbf{D}_\delta(\mathbf{z}^t)$, the probability of $\mathbf{z}^{t+1}$ = $\mathbf{c}$ is not zero.)

Theorem 4.1. *If GSA satisfies the condition C1 and C2, it is globally convergent:*

$$\lim_{t\to\infty} P\left(\mathbf{p}^t = \mathbf{o}_G\right) = 1.$$

Proof. This proof is based on the proof in [144]. From the condition C1, if $\mathbf{p}^t$ is a global minimizer $\mathbf{o}_G$, $\mathbf{p}^{t+t'}$ for $t' > 0$ is also the global minimizer $\mathbf{o}_G$. Therefore, if $\mathbf{p}^t \neq \mathbf{o}_G$, then $\mathbf{p}^i \neq \mathbf{o}_G$ for $\forall i < t$.

$$\begin{aligned} P\left(\mathbf{p}^t = \mathbf{o}_G\right) &= 1 - P\left(\mathbf{p}^t \neq \mathbf{o}_G\right) \\ &\geq 1 - \prod_{i=1}^{t-1}\left(1 - P\left(\mathbf{p}^i = \mathbf{o}_G\right)\right) \end{aligned}$$

Since $P\left(\mathbf{z}^i = \mathbf{o}_G\right) > 0$ for $\forall i \in \mathcal{I}$ by the condition C2, $P\left(\mathbf{p}^i = \mathbf{o}_G\right) > 0$ for $\forall i \in \mathcal{I}$. Therefore,

$$\lim_{t=1}^{\infty}\prod_{i=1}^{t-1}\left(1 - P\left(\mathbf{p}^i = \mathbf{o}_G\right)\right) \leq \lim_{t=1}^{\infty}\prod_{i\in\mathcal{I}}^{t-1}\left(1 - P\left(\mathbf{p}^i = \mathbf{o}_G\right)\right) = 0$$

and $\lim_{t\to\infty} P\left(\mathbf{p}^t = \mathbf{o}_G\right) = 1$. □

Theorem 4.2. *If GSA satisfies the condition C1 and C3, it is locally convergent:*

$$\lim_{t\to\infty} P\left(\mathbf{p}^t = \mathbf{o}_L\right) = 1.$$

Proof. First of all, by the assumption $\delta \geq \delta^*$, $\mathbf{D}_\delta(\mathbf{z}^t)$ includes both adjacent points of the centre $\mathbf{z}^t$ along each axis, by Condition C3 the sequence

$$\left(\mathbf{z}^t, \mathbf{z}^{t+1}, \ldots\right)$$

can arrive any point in the space with non-zero probability.

Let $\mathbf{c}_1^1$ be a minimizer in $\mathbf{D}_\delta(\mathbf{z}^{i_1})$. If $\mathbf{c}_1^1$ is not a local minimizer in $\mathbf{D}_\delta(\mathbf{c}_1^1)$, let $\mathbf{c}_1^2$ be a minimizer in $\mathbf{D}_\delta(\mathbf{c}_1^1)$. We repeat the process for $\mathbf{D}_\delta(\mathbf{c}_1^1), \mathbf{D}_\delta(\mathbf{c}_1^2), \ldots$ and eventually find $k_1 \geq 1$ such that $\mathbf{c}_1^{k_1}$ is a local minimizer in $\mathbf{D}_\delta(\mathbf{c}_1^{k_1})$. Therefore, $\mathbf{c}_1^{k_1}$ is a minimizer in

$$\mathbf{D}_1 = \mathbf{D}_\delta(\mathbf{z}^{i_1}) \cup \mathbf{D}_\delta(\mathbf{c}_1^1) \cup \cdots \cup \mathbf{D}_\delta(\mathbf{c}_1^{k_1}).$$

Let's consider the following assumption:

- Assumption $\mathcal{A}_1$: A subset of random samples $\{\mathbf{z}^t, t = 1, \ldots\}$ are generated infinitely many times in the set $\mathbf{D}_1$ and not generated infinitely many times in the set $\mathbf{D} \setminus \mathbf{D}_1$.

If $\mathcal{A}_1$ holds, let $\mathcal{I}_1 \subset \mathbb{N}^+$ be the set of indexes with infinite elements such that $\forall t \in \mathcal{I}_1, \mathbf{z}^t \in \mathbf{D}_1$. A point $\mathbf{z}^t, t \in \mathcal{I}_1$ can arrive any points in $\mathbf{D}_1$ after $k_1 + 1$ step with non-zero probability. Therefore there exists a subset $\mathcal{I}_{s1} \subset \mathcal{I}_1$ such that

$$\forall t \in \mathcal{I}_{s1} \ \ \forall \mathbf{c} \in \mathbf{D}_1 \qquad P\left(\mathbf{z}^t = \mathbf{c}\right) > 0$$

By Theorem 4.1, $\mathbf{p}^t$ for $t = 1, \ldots$ will converge to the minimizer $\mathbf{c}_1^{k_1}$ in $\mathbf{D}_1$. If $\mathcal{A}_1$ does not hold, we select a random sample $\mathbf{z}^{i_2}$ from $\mathbf{D} \setminus \mathbf{D}_1$. Let $\mathbf{c}_2^1$ be a minimizer in $\mathbf{D}_\delta(\mathbf{z}^{i_2})$. Repeating the same process, we obtain a subset

$$\mathbf{D}_2 = \mathbf{D}_\delta(\mathbf{z}^{i_2}) \cup \mathbf{D}_\delta(\mathbf{c}_2^1) \cup \cdots \cup \mathbf{D}_\delta(\mathbf{c}_2^{k_2}).$$

and $\mathbf{c}_2^{k_2}$ is a minimizer in $\mathbf{D}_2$. If a subset of random samples $\{\mathbf{z}^t, t = 1, \ldots\}$ are generated infinitely many times in the subset $\mathbf{D}_1 \cup \mathbf{D}_2$ and not generated infinitely many times in the set $\mathbf{D} \setminus (\mathbf{D}_1 \cup \mathbf{D}_2)$, $\mathbf{p}^t, t = 1, \ldots$ will converge to the minimizer $\min(\mathbf{c}_1^{k_1}, \mathbf{c}_2^{k_2})$ in $\mathbf{D}_1 \cup \mathbf{D}_2$. Otherwise we select a random sample $\mathbf{z}^{i_3}$ from $\mathbf{D} \setminus (\mathbf{D}_1 \cup \mathbf{D}_2)$. Since $\mathbf{D}_2$ includes a point $\mathbf{z}^{i_2}$ which is not in $\mathbf{D}_1$, $|\mathbf{D}_1 \cup \mathbf{D}_2| > |\mathbf{D}_1|$. Therefore we will eventually obtain a number l such that a subset of random samples $\{\mathbf{z}^t, t = 1, \ldots\}$ are generated infinitely many times in the subset $\cup_{i=1}^{l} \mathbf{D}_i$ and not generated infinitely many times in the set $\mathbf{D} \setminus \cup_{i=1}^{l} \mathbf{D}_i$, because $\mathbf{D}$ is a finite space. Thus $\mathbf{p}^t, t = 1, \ldots$ will converge to a local minimum in $\cup_{i=1}^{l} \mathbf{D}_i$. □

From the update rule of pbest (4.11), the general model in Section 4.3.1 is a realization of GSA.

4.3.3 Discrete PSO derived from SPSO (B(D)-SPSO)

A discrete version of SPSO in Section 2.5.2 can be derived using only basic components described in Section 4.3.1. We denote the binary version of SPSO as B-SPSO, the discrete version with the round functions as D-SPSO. For the binary PSO (B-SPSO), however, we add the following perturbation parameter ξ:

$$\psi_j(x_j) = \begin{cases} \xi & \text{if } x_j < \xi \\ 1 - \xi & \text{if } x_j > 1 - \xi \end{cases} \tag{4.17}$$

where ξ is a positive constant $\xi > 0$. Without this kind of perturbation, the particles converge directly to the optimal point without exploring the points in the neighborhood as shown in the experiments. As a result we need to take care of the additional parameter ξ in B-SPSO. In general it is desirable to adjust the size of ξ so that we can explore the search space in the early stages of the optimization process and exploit solutions around the optimal point in the later stages of the optimization process.

By the perturbation parameter ξ the condition C3 is satisfied. (The probability of moving to any points in the neighborhood of the pbest is non-zero.) Therefore B-SPSO is locally convergent.

4.3.4 Discrete PSO derived from LcRiPSO (B(D)-SLRPSO, B(D)-FLRPSO)

In this section we derive the discrete versions of PSO from LcRiPSO in Section 4.2.2. One is derived from SLcRiPSO with the gbest topology and the other is derived from FLcRiPSO with the ring topology. We denote the former as B-SLRPSO for the binary version and D-SLRPSO for the discrete version with the round functions and the latter as B-FLRPSO and D-FLRPSO. The velocity update rule for B(D)-SLRPSO is defined as follows:

$$\begin{aligned}\mathbf{v}_i^{t+1} =& \omega \mathbf{v}_i^t + c_1 r_{1,i}^t \left(\psi(N\left(\mathbf{p}_i^t, \sigma^2 I\right)) - \mathbf{x}_i^t\right) \\ &+ c_2 r_{2,i}^t \left(\psi(N\left(\mathbf{g}^t, \sigma^2 I\right)) - \mathbf{x}_i^t\right)\end{aligned} \tag{4.18}$$

where $N\left(\mathbf{p}_i^t, \sigma^2 I\right)$ is a random vector which is generated from the normal distribution with the mean vector $\mathbf{p}_i^t$ and the covariance matrix $\sigma^2 I$. ψ is a discretization function in (4.15) or (4.16).

The velocity update rule for B(D)-FLRPSO is defined as follows:

$$\begin{aligned}\mathbf{v}_i^{t+1} =& \omega \mathbf{v}_i^t + \frac{c_1 r_{1,i}^t}{3} \left(\psi(N\left(\mathbf{p}_i^t, \sigma^2 I\right)) - \mathbf{x}_i^t\right) \\ &+ \frac{c_2 r_{2,i}^t}{3} \left(\psi(N\left(\mathbf{p}_{i-1}^t, \sigma^2 I\right)) - \mathbf{x}_i^t\right) \\ &+ \frac{c_3 r_{3,i}^t}{3} \left(\psi(N\left(\mathbf{p}_{i+1}^t, \sigma^2 I\right)) - \mathbf{x}_i^t\right)\end{aligned} \tag{4.19}$$

where we use the ring topology which refers to its own pbest and the pbests of two adjacent particles for the velocity update.

By the perturbation function $N\left(\mathbf{p}_i^t, \sigma^2 I\right)$, the condition C3 is satisfied. (The probability of moving to any points in the neighborhood of the pbest is non-zero.) Therefore B(D)-SLRPSO and B(D)-FLRPSO are locally convergent.

We check the stability conditions for these methods.

Lemma 4.1. *The stability conditions for SPSO are applied to B(D)-SLRPSO without any changes. The stability conditions for FIPS are applied to B(D)-FLRPSO without any changes.*

Proof. The stability conditions (4.3) for SPSO are derived assuming

$$x^{t+1} = lx^t - \omega x^{t-1} + \phi_1 p + \phi_2 g$$

where $l = \omega + 1 - \phi_1 - \phi_2$ and $\omega, \phi_1, \phi_2, p, g$ are random numbers. In B(D)-SLRPSO, $\mathbf{p}_i^t$ is replaced by $N(\mathbf{p}_i^t, \sigma^2 I)$ which can also be modeled as a random number p. Since the derived conditions are independent of the value of p, the stability conditions for B(D)-SLRPSO of (4.18) are the same as those for SPSO. Similarly, the stability conditions for B(D)-FLRPSO of (4.19) are the same as those for FIPS. □

B(D)-SLRPSO and B(D)-FLRPSO inherit the properties of LcRiPSO and solve the stagnation problem, the small size problem and the scale problem in Section 4.2.2. B(D)-SLRPSO and B(D)-FLRPSO are also "approximately" rotationally invariant. To see this let us consider D-SLRPSO and D-FLRPSO with the round functions. According to the theorem in Appendix 2 in [14], in order for a function f which maps $\mathbb{R}^{d'} \Rightarrow \mathbb{R}^{d'}$ to be rotationally invariant, f needs to satisfy the the condition (4.7). Therefore, the function ψ in (4.18) also needs to satisfy the condition (4.7). For the round function in (4.16), domain spaces are divided into hyper-cubes in which all points in a cube are evaluated at the same point. After the rotation operation the hyper-cube is rotated around the origin. In the rotated space, however, we use the same evaluation points along each axis. Therefore we will obtain different results if we apply the round function on the rotated space or if we first apply the round function and rotate the space, because the hyper-cube is rotated (takes different shape) and the evaluation points along each axis are shifted after the rotation (Figure 4.2). However, the deviations of the hyper-cubes are not so large and the rotationally invariant property is approximately preserved. We will check it in the experiments in Section 4.4.6.

4.3.5 Specification of Perturbation Parameters

We introduced the perturbation parameter ξ for B-SPSO in (4.17) in Section 4.3.3 and σ for B(D)-SLRPSO and B(D)-FLRPSO in (4.19) and (4.19) in Section 4.3.4. The best specifications of those parameters vary according to the dimension of spaces and the size of population ([14]). Therefore it is not an easy task to find the optimal specification of those parameters. In this section we propose a new method to ease this process of finding the optimal σ for B(D)-SLRPSO and B(D)-FLRPSO.

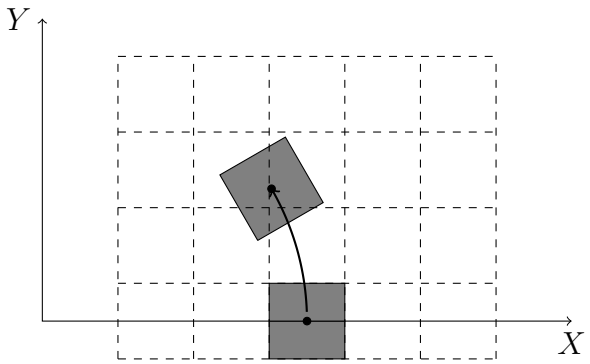


Figure 4.2: Rotation of XY coordinate system

First of all let us start our consideration by visualizing how the optimal values of ξ and σ vary according to the different dimension of spaces and the different sizes of population. We use the benchmark function F2 and F6 in Table 4.1. Figure 4.3 are contour maps of optimal solutions in which Y-axis is the perturbation parameter $\xi \in \{0.01, 0.05, 0.1, 0.15, 0.2\}$ of B-SPSO and X-axis is the size of dimension $\in \{20, 80, 140, 200\}$ with the fixed population $= 200$. Figure 4.4 are contour maps of optimal solutions in which Y-axis is the perturbation parameter $\xi \in \{0.01, 0.05, 0.1, 0.15, 0.2\}$ of B-SPSO and X-axis is the size of population $\in \{100, 200, 500, 1000\}$ with the fixed dimension $= 80$. For each point we repeat the experiments three times and compute the average of the results. The left figure is the one for F2 and the right figure is the one for F6.

Similarly Figure 4.5 - Figure 4.8 show the contour maps of the optimal solutions for the input space of the perturbation parameter $\log_{1/10}\sigma \in \{1, 2, 3, 4, 5, 6\}$ vs. the size of dimension $\in \{20, 80, 140, 200\}$ and the perturbation parameter vs. the size of population $\in \{100, 200, 500, 1000\}$ for B-SLRPSO and B-FLRPSO.

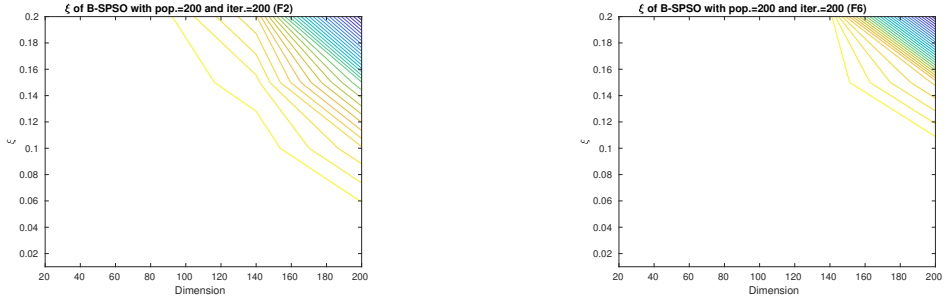
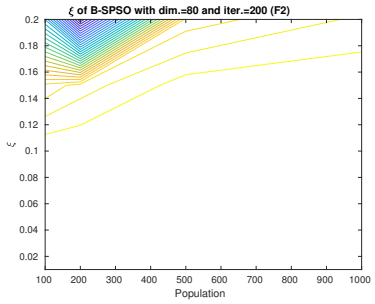


Figure 4.3: ξ of B-SPSO vs. Dimension of Space

We introduce a new method to find the optimal specification of σ for B(D)-SLRPSO, B(D)-FLRPSO and LcRiPSO in Section 4.2.2. It makes use of the "difference" of pbest



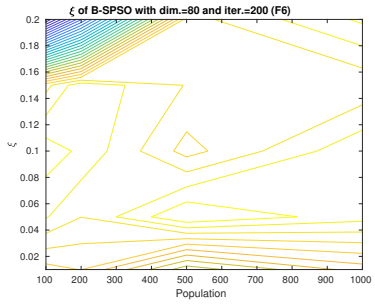
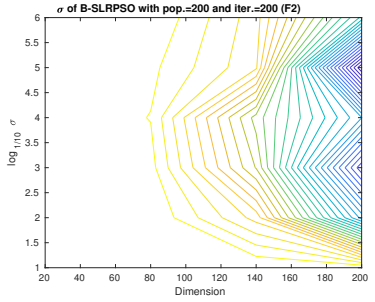


Figure 4.4: ξ of B-SPSO vs. Size of Population



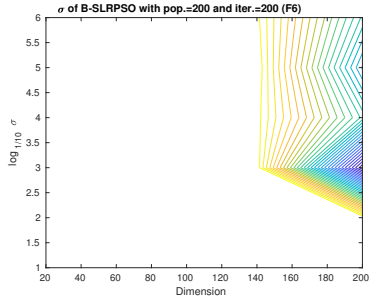
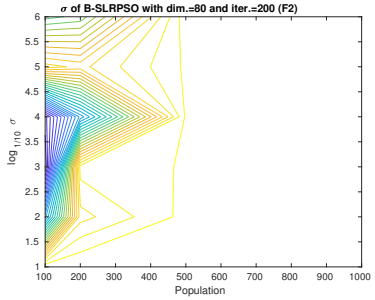


Figure 4.5: σ of B-SLRPSO vs. Dimension of Space



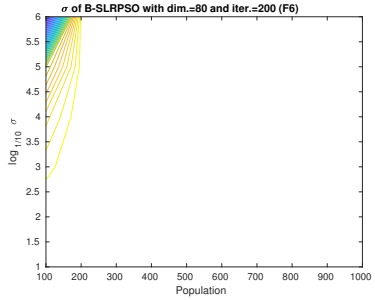
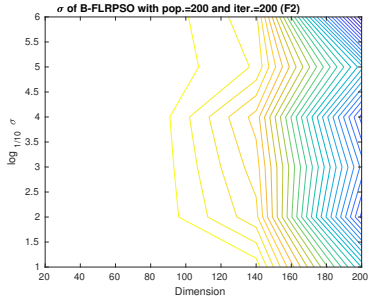


Figure 4.6: σ of B-SLRPSO vs. Size of Population



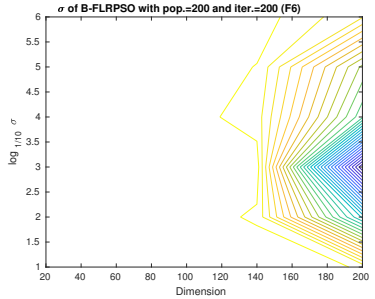


Figure 4.7: σ of B-FLRPSO vs. Dimension of Space

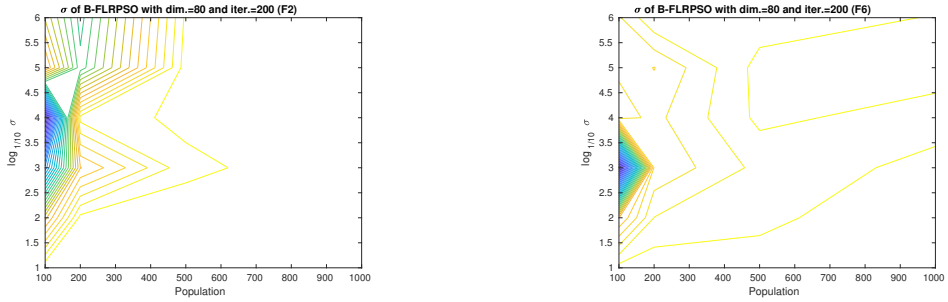


Figure 4.8: σ of B-FLRPSO vs. Size of Population

before and after the perturbation. For binary PSO we define the "difference" as the ratio of unchanged bits after the perturbation. We denote the ratio as RP (ratio of preservation).

$$RP := \frac{\text{unchanged bits after the perturbation}}{\text{number of bits}} \tag{4.20}$$

In Figure 4.25, 7 bits out of 10 are unchanged after the perturbation. In this case, $RP = 0.7$.

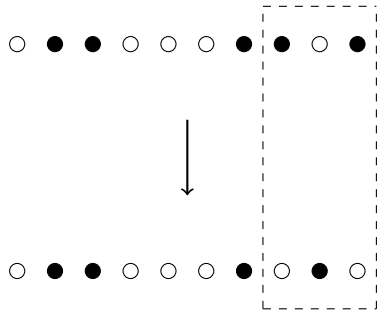


Figure 4.9: Ratio of Preservation (RP)

For continuous PSO, we define the "difference" as the ratio of the average distance of pbest moved by the perturbation operation to the radius of the whole space. We denote the ratio as RPD (ratio of perturbed distance).

$$RPD := \frac{\text{average distance of pbest moved by perturbation operation}}{\text{radius of whole space}} \tag{4.21}$$

Suppose that $pbest$ is the position of pbest and $pbest2$ is the position of pbest after the perturbation. Assuming $u_j - l_j$ in (2.29) is a constant c for each dimension $j = 1, \ldots, d'$, RPD is computed as:

$$RPD = \frac{\text{mean}(\|pbest2 - pbest\|)}{\sqrt{d'}c}. \tag{4.22}$$

where $\|\cdot\|$ is a Euclidean norm.

Figure 4.10 - Figure 4.13 are the contour maps of the optimal solutions for the input space of the ratio of preservation $\in \{0.60, 0.70, 0.80, 0.85, 0.90, 0.95, 0.99\}$ vs. the size of dimension $\in \{20, 80, 140, 200\}$ and the ratio of preservation vs. the size of population $\in \{100, 200, 500, 1000\}$ for B-SLRPSO and B-FLRPSO.

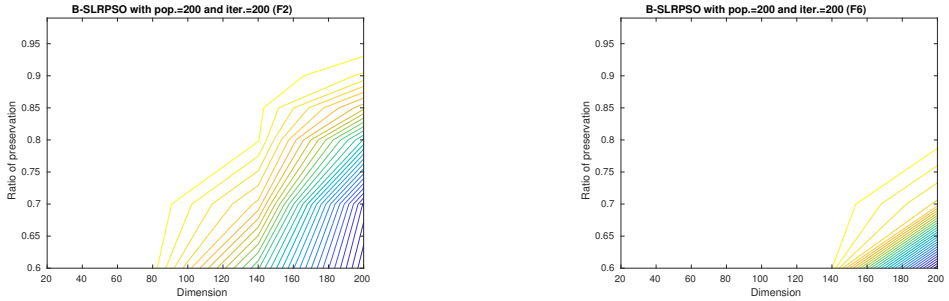


Figure 4.10: Ratio of Preservation of B-SLRPSO vs. Dimension of Space

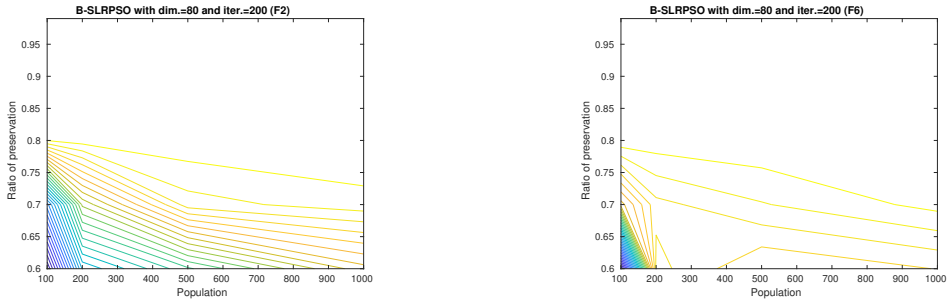


Figure 4.11: Ratio of Preservation of B-SLRPSO vs. Size of Population

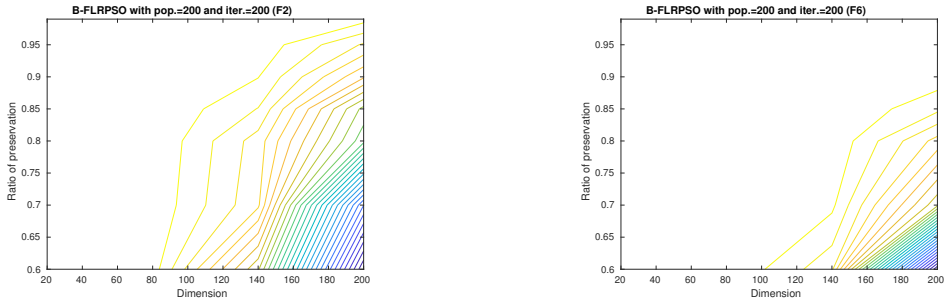


Figure 4.12: Ratio of Preservation of B-FLRPSO vs. Dimension of Space

The advantages of the proposed method are as follows:

- As shown in Figure 4.10 - Figure 4.13 the relation between RP and the dimension sizes or the population sizes is more stable and consistent compared to Figure 4.5 - Figure 4.8.

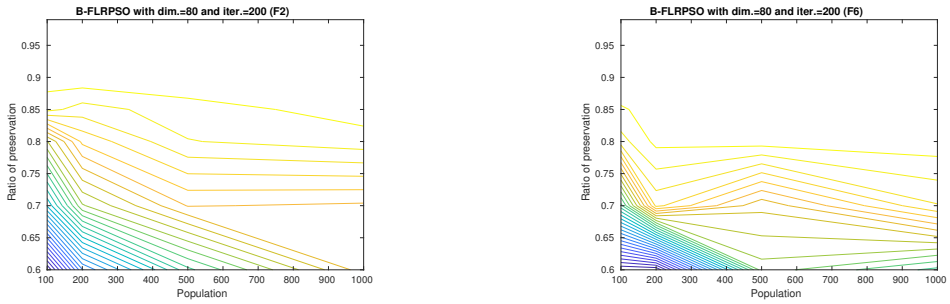


Figure 4.13: Ratio of Preservation of B-FLRPSO vs. Size of Population

- In the case of binary PSO we can restrict the search space in $[0, 1]$. For the continuous PSO the search space is also more predictable than directly searching for the optimal σ.

We also verify those points through the examples in the experiments.

We can write a program which automatically determine the value of σ which achieves the specified value of RP or RPD at the first iteration. The details of this procedure can be found in Algorithm 4.1. Firstly we determine the range which includes the target value using a rough step ζ. Then we apply the binary search to the region until we approach the target value within the specified distance ϵ.

4.4 Experiments

In this section, we conduct a number of experiments to verify the performance and properties of the proposed discrete PSO. The main objectives of the experiments are summarized as follows.
The main objectives:

- For the binary PSO, we compare the performance of the "position as probability" approach with that of the "velocity as probability" approach using benchmark functions and Knapsack Problems.

- For the discrete PSO, we examine the performance of the proposed methods with the round functions using Independent Job Scheduling Problems and examine the possibility of the efficacy improvement using the cache method without degrading quality of solutions.

We also conduct experiments to compare the Gray encoding with the binary encoding.

Algorithm 4.1: Determination of σ which achieves the specified RP or RPD

Require: pbest, Starting value of σ_0, Target value T of (RP or RPD), Rough step ζ, Threshold size ϵ

Step 1: Determination of the range which includes T;

Compute the new pbest by applying the perturbation with σ_0

Compute the difference T_0 between the new pbest and the original pbest

$k = 0$

while $true$ **do**

$\sigma_{k+1} = \sigma_k \pm \zeta$

(We increase or decrease the value of σ_k by ζ so that the difference T_k approaches to T.)

if T_{k+1} exceeds T (if $sign(T_{k+1} - T) = -sign(T_k - T)$) **then**

Break;

end if

$k = k + 1$

end while

Step 2: Binary Search;

Suppose that $T_{k+1} > T_k$ ($T_k > T_{k+1}$)

Set $high = \sigma_{k+1}$, $low = \sigma_k$ (Set $high = \sigma_k$, $low = \sigma_{k+1}$)

while $\|T_{k+1} - T_k\| > \epsilon$ **do**

$k = k + 1$;

Compute the difference T_{k+1} using $\sigma_{k+1} = \frac{high+low}{2}$

if $T_{k+1} > T + \epsilon$ **then**

Set $high = \sigma_{k+1}$

else if $T_{k+1} < T - \epsilon$ **then**

Set $low = \sigma_{k+1}$

else

Break;

end if

end while

return σ_{k+1}

In this chapter, we use four basic functions; Sphere, Rosenbrock, Griewangk and Rastrigin and the CEC 2005 benchmark functions in Suganthan et al. [150] to test the new PSO methods. Table 4.1 lists those functions.

4.4.1 Experiments for Binary PSO using Benchmark Functions

In this section we conduct the experiments of binary PSO using benchmark functions. We start with the preliminary experiments to determine the parameter (ω, c_1, c_2) and the perturbation parameters ξ and σ.

Preliminary Experiments

Specification of $(\omega, c_1, c_2), (\omega, c_1, c_2, c_3)$: As stated in Section 4.2.1 the condition for the order-2 stability of SPSO is given as

$$-1 < \omega < 1, \qquad 0 < c_1 + c_2 < 24\frac{1-\omega^2}{7-5\omega}.$$

According to this conditions, assuming that $c_1 = c_2$ each pair of $(\omega, c_k(k = 1, 2))$ must fall in the region depicted in Figure 4.14.

Similarly the order-2 stability condition for FIPS is given as

$$-1 < \omega < 1, \qquad 0 < c_1 + \ldots + c_\kappa < \frac{12\kappa\,(1-\omega^2)}{3\kappa + 1 + \omega\,(1-3\kappa)}. \tag{4.23}$$

where κ is the number of neighborhood of each particle. In the case of B-FLRPSO with the ring topology, κ is three for all particles. Assuming that $c_1 = c_2 = c_3$, each pair of $(\omega, c_k(k = 1, 2, 3))$ must fall in the region depicted in Figure 4.15.

In this section we examine the performance of different sets of (ω, c_1, c_2) and (ω, c_1, c_2, c_3) chosen from the order-2 stability region. In the experiments we randomly select eight points from the region which are shown in Figure 4.14 and Figure 4.15. We compare the performance on those points using the benchmark function F1, F2, F4, F6 and F9 in Section 1.6. We set the dimension of real solutions to $3, 6, 9$. Each real number is represented by 20 bits of binary code and the total number of the dimensions are $60, 120, 180$, respectively. We set the number of iteration and the size of population to 200. We repeat the experiments 30 times.

Table 4.2 and Table 4.3 show the results. In each row we emphasize the best result in boldface. There are no clear differences among those specifications. We will use the setting of the left-most column of the tables in the following experiments.

Table 4.1: Four Basic Functions and CEC 2005 Benchmark Functions [150]

ID	Formulation	Domain	Mode
S1	$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2$	$[-50, 50]$	Unimodal
S2	$f(\mathbf{x}) = \sum_{i=1}^{d} [100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2]$	$[-50, 50]$	Multi-modal
S3	$f(\mathbf{x}) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos \frac{x_i}{\sqrt{i}} + 1$	$[-50, 50]$	Multi-modal
S4	$f(\mathbf{x}) = \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-50, 50]$	Multi-modal
F1	$f(\mathbf{x}) = \sum_{i=1}^{d} (z_i)^2 - 450$	$[-100, 100]$	Unimodal
	$\mathbf{z} = \mathbf{x} - \mathbf{o}$.		
	$\mathbf{o}$ is a randomly generated vector in the domain of $\mathbf{x}$		
F2	$f(\mathbf{x}) = \sum_{i=1}^{d} (\sum_{j=1}^{i} z_j)^2 - 450$	$[-100, 100]$	Unimodal
	$\mathbf{z} = \mathbf{x} - \mathbf{o}$		
F4	$f(\mathbf{x}) = \sum_{i=1}^{d} (\sum_{j=1}^{i} z_j)^2 \times (1 + 0.4\lvert N(0,1)\rvert) - 450$	$[-100, 100]$	Unimodal
	$\mathbf{z} = \mathbf{x} - \mathbf{o}$		
	$N(0,1)$ is a randomly generated number from the standard normal distribution		
F5	$f(\mathbf{x}) = \max_i \lvert\mathbf{A}_i\mathbf{x} + \mathbf{B}_i\rvert - 310$	$[-100, 100]$	Unimodal
	$\mathbf{A}_i$ is i-th row of a $d \times d$ matrix $\mathbf{A}$ ($\det(\mathbf{A}) \neq 0$), $\mathbf{B}_i = \mathbf{A}_i \times \mathbf{o}$		
	An element a_{ij} in $\mathbf{A}$ is a randomly generated number in $[-500, 500]$		
F6	$f(\mathbf{x}) = \sum_{i=1}^{d-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + 390$	$[-100, 100]$	Multi-modal
	$\mathbf{z} = \mathbf{x} - \mathbf{o} + 1$		
F9	$f(\mathbf{x}) = \sum_{i=1}^{d} (z_i^2 - 10\cos(2\pi z_i) + 10) - 330$	$[-5, 5]$	Multi-modal
	$\mathbf{z} = \mathbf{x} - \mathbf{o}$		
F10	$f(\mathbf{x}) = \sum_{i=1}^{d} (z_i^2 - 10\cos(2\pi z_i) + 10) - 330$	$[-5, 5]$	Multi-modal
	$\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}$		
	$\mathbf{M}$ is a linear transformation matrix with condition number 2.		
F11	$f(\mathbf{x}) = \sum_{i=1}^{d} (\sum_{k=0}^{20} [0.5^k \cos(2\pi 3^k (z_i + 0.5))]) - d\sum_{k=0}^{20} [0.5^k cos(2\pi 3^k * 0.5)] + 90$	$[-0.5, 0.5]$	
	$\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}$		
	$\mathbf{M}$ is a linear transformation matrix with condition number 5.		
F12	$f(\mathbf{x}) = \sum_{i=1}^{d} (\mathbf{A}_i - \mathbf{B}_i(\mathbf{x}))^2 - 460$	$[-100, 100]$	Multi-modal
	$\mathbf{A}_i = \sum_{j=1}^{d} (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j), \mathbf{B}_i(\mathbf{x}) = \sum_{j=1}^{d} (a_{ij} \sin x_j + b_{ij} \cos x_j)$		
	a_{ij}, b_{ij} are integer random number in the range $[-100, 100]$.		
	$\boldsymbol{\alpha}$ are random numbers in the range $[-\pi, \pi]$.		
F14	$f(\mathbf{x}) = F(z_1, z_2) + F(z_2, z_3) + \cdots + F(z_d, z_1) - 300$	$[-100, 100]$	Multi-modal
	$F(x, y) = 0.5 + \frac{\sin^2(\sqrt{x^2+y^2}) - 0.5}{(1+0.001(x^2+y^2))^2}$		
	$\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M}$		
	$\mathbf{M}$ is a linear transformation matrix with condition number 3.		

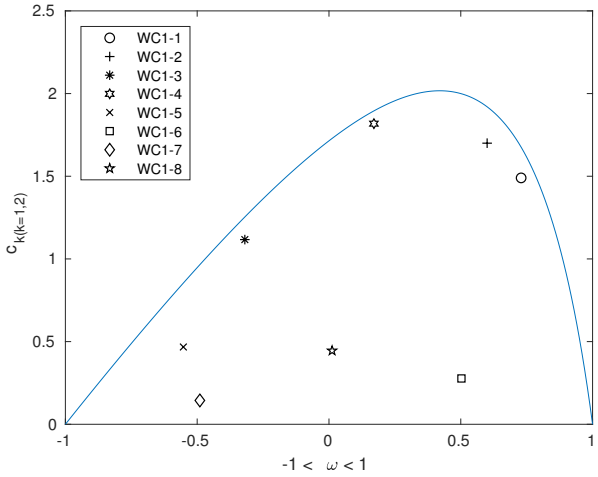


Figure 4.14: The convergence region of $(\omega, c_{k(k=1,2)})$ for SPSO

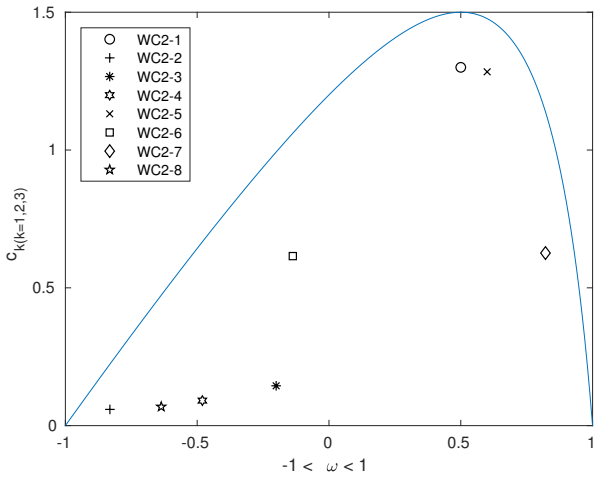


Figure 4.15: The convergence region of $(\omega, c_{k(k=1,2,3)})$ for FIPS

Table 4.2: Experiments for B-SPSO with different $(\omega, c_{k(k=1,2)})$

Func.	Dim.	**WC1-1** $\omega = 0.729$ $c_k = 1.49$	**WC1-2** $\omega = 0.6$ $c_k = 1.7$	**WC1-3** $\omega = -0.319$ $c_k = 1.116$	**WC1-4** $\omega = 0.170$ $c_k = 1.818$	**WC1-5** $\omega = -0.552$ $c_k = 0.467$	**WC1-6** $\omega = 0.502$ $c_k = 0.277$	**WC1-7** $\omega = -0.489$ $c_k = 0.144$	**WC1-8** $\omega = 0.011$ $c_k = 0.445$
F1	3	**4.21e-09**	**4.21e-09**	**4.21e-09**	**4.21e-09**	7.12e-09	**4.21e-09**	1.06e-05	**4.21e-09**
	6	2.02e-08	**1.82e-08**	8.33e-08	1.98e-08	3.07e-06	7.67e-07	0.000373	3.74e-07
	9	5.65e-06	**2.73e-06**	1.2e-05	4.65e-06	9.34e-05	3.22e-05	0.00474	3.77e-05
F2	3	**7.58e-08**	1.12e-07	3.74e-07	2.1e-07	4.54e-06	7.06e-06	0.00116	3.32e-06
	6	0.000879	**0.000501**	0.000861	0.000519	0.00397	0.00304	0.0946	0.00174
	9	0.0361	0.0402	0.0612	**0.0292**	0.116	0.0797	1.07	0.131
F4	3	3.14e-06	**2.29e-06**	1.58e-05	3.92e-06	0.000592	5.53e-05	0.00198	2.12e-05
	6	0.00552	**0.00413**	0.0189	0.0159	0.0212	0.0191	0.164	0.0106
	9	0.307	**0.197**	0.522	0.342	1.02	0.545	4.17	0.646
F6	3	4.3	9.48	6.03	8.33	**3.48**	4.52	3.93	5.38
	6	**5.08**	12	6.8	6.19	10.8	5.49	18.5	8.18
	9	10.5	**7.08**	35.3	25.2	26.7	8.95	21.9	13
F9	3	0.0663	0.133	0.0332	0.0332	0.0663	0.0663	**3.87e-09**	0.0332
	6	0.398	0.464	0.564	0.564	**0.332**	0.431	0.464	0.398
	9	1.43	1.43	1.66	**1.23**	1.33	1.23	1.42	1.33

Table 4.3: Experiments for B-FLRPSO with different $(\omega, c_{k(k=1,2,3)})$

Func.	Dim.	**WC2-1** $\omega = 0.5$ $c_k = 1.3$	**WC2-2** $\omega = -0.831$ $c_k = 0.058$	**WC2-3** $\omega = -0.200$ $c_k = 0.144$	**WC2-4** $\omega = -0.480$ $c_k = 0.090$	**WC2-5** $\omega = 0.600$ $c_k = 1.283$	**WC2-6** $\omega = -0.137$ $c_k = 0.615$	**WC2-7** $\omega = 0.821$ $c_k = 0.626$	**WC2-8** $\omega = -0.636$ $c_k = 0.068$
F1	3	**5.8e-06**	6.58e-06	4.69e-05	2.28e-05	0.000128	0.000123	0.003	6.45e-05
	6	**2.53e-06**	3.75e-06	0.000364	1.84e-05	0.000267	6.58e-05	0.0121	9.01e-05
	9	**6.64e-06**	1.18e-05	0.00175	3.48e-05	0.000406	4.73e-05	0.0295	8.02e-05
F2	3	0.267	0.201	**0.0125**	0.239	0.137	0.473	0.18	0.371
	6	3.08	5.61	**2.09**	3.85	5.21	8.38	9.78	10.7
	9	24.8	23.1	**18.7**	28.8	32.6	49.7	51.5	38.7
F4	3	0.543	0.272	**0.081**	0.287	0.115	0.468	0.167	1.03
	6	17.3	18	**6.65**	12.9	10.4	21.1	12.1	23
	9	98.4	85.7	**62.7**	85.9	91.7	142	108	106
F6	3	0.152	0.154	0.0159	0.181	**0.00873**	0.132	0.0186	0.0718
	6	0.18	0.202	**0.111**	0.197	0.192	0.261	0.824	0.197
	9	0.51	**0.492**	1.33	0.759	1.37	0.887	5.04	1.57
F9	3	**3.87e-09**	**3.87e-09**	**3.87e-09**	**3.87e-09**	**3.87e-09**	**3.87e-09**	**3.87e-09**	**3.87e-09**
	6	**9.45e-09**	0.0332	2.34e-08	9.58e-09	0.000109	6.49e-08	0.000414	1.63e-06
	9	0.48	0.569	0.437	0.937	0.269	**0.0163**	0.373	0.213

Specification of Perturbation Parameters: In Section 4.3.5 we examined the effects of the various sizes of perturbation parameters according to the different sizes of di-

mension and population. As the next task we determine the best combination of perturbation parameters. In the following experiments we adopt the adaptive scheme of the parameter specification such that particles explore the wider range at the beginning of the iteration and investigate more closely around the best points at the end of the iteration.

Based on Figure 4.3 and Figure 4.4, we set the range

$$\xi \in \{0.01, 0.03, 0.05, 0.07, 0.09, 0.11, 0.13, 0.15\} \tag{4.24}$$

and decide the best pair of (ξ_{start}, ξ_{end}) from this range under the condition that $\xi_{start} \geq \xi_{end}$. We evaluate each pair of ξ in (4.28) using four benchmark functions F2, F4, F6 and F9. We set the dimension to $80(4 \times 20)$ and the number of iteration and the size of population to 300. For each pair we repeat the experiments three times and compute the average.

Table 4.4 - Table 4.7 show the rank of the pairs of (ξ_{start}, ξ_{end}) for each function. The smaller ranks mean better results. Table 4.8 is the summation of those ranks. The pairs $\{(0.11, 0.01)$, $(0.09, 0.01)$, $(0.07, 0.01)$, $(0.05, 0.01)$ $\}$ seem to be better choices than others. From the range we select the pair $(0.08, 0.01)$ for the following experiments.

Table 4.4: Pair of (ξ_{start}, ξ_{end}) for F2

$\xi_{start} \backslash \xi_{end}$	0.15	0.13	0.11	0.09	0.07	0.05	0.03	0.01
0.15	36	34	33	32	25	23	16	11
0.13	-	35	31	29	27	19	15	10
0.11	-	-	30	28	24	21	13	9
0.09	-	-	-	26	20	18	17	1
0.07	-	-	-	-	22	14	8	5
0.05	-	-	-	-	-	12	7	2
0.03	-	-	-	-	-	-	6	3
0.01	-	-	-	-	-	-	-	4

Based on Figure 4.10 - Figure 4.13 we set the range of the ratio of preservation (RP) for B-SLRPSO and B-FLRPSO as follows.

$$\{0.75, 0.80, 0.85, 0.90, 0.95, 0.99\}. \tag{4.25}$$

We conduct the same experiments as for ξ and choose the best pair of (RP_{start}, RP_{end}) for B-SLRPSO and B-FLRPSO.

Table 4.5: Pair of (ξ_{start}, ξ_{end}) for F4

$\xi_{start} \backslash \xi_{end}$	0.15	0.13	0.11	0.09	0.07	0.05	0.03	0.01
0.15	36	34	32	30	28	22	14	16
0.13	-	35	33	29	26	17	18	6
0.11	-	-	31	27	20	19	13	15
0.09	-	-	-	23	21	11	9	4
0.07	-	-	-	-	24	12	8	2
0.05	-	-	-	-	-	10	5	1
0.03	-	-	-	-	-	-	7	3
0.01	-	-	-	-	-	-	-	25

Table 4.6: Pair of (ξ_{start}, ξ_{end}) for F6

$\xi_{start} \backslash \xi_{end}$	0.15	0.13	0.11	0.09	0.07	0.05	0.03	0.01
0.15	32	17	12	26	18	31	27	30
0.13	-	21	28	14	11	24	2	29
0.11	-	-	7	10	36	5	20	9
0.09	-	-	-	13	22	25	34	23
0.07	-	-	-	-	15	4	16	3
0.05	-	-	-	-	-	6	8	1
0.03	-	-	-	-	-	-	33	35
0.01	-	-	-	-	-	-	-	19

Table 4.7: Pair of (ξ_{start}, ξ_{end}) for F9

$\xi_{start} \backslash \xi_{end}$	0.15	0.13	0.11	0.09	0.07	0.05	0.03	0.01
0.15	29	19	18	5	28	4	2	20
0.13	-	27	16	6	14	26	3	7
0.11	-	-	15	17	25	33	8	9
0.09	-	-	-	13	30	21	31	10
0.07	-	-	-	-	22	23	32	24
0.05	-	-	-	-	-	1	11	34
0.03	-	-	-	-	-	-	12	36
0.01	-	-	-	-	-	-	-	35

Table 4.9 - Table 4.12 show the rank of the pairs of (RP_{start}, RP_{end}) of B-SLRPSO for each function. Table 4.13 is the summation of those ranks. The pairs $\{(0.85, 0.99)$, $(0.90, 0.99)$, $(0.95, 0.99)$ $\}$ seem to be better choices than others. We will use the pair

Table 4.8: Pair of (ξ_{start}, ξ_{end}) for Sum of (F2, F4, F6, F9)

$\xi_{start} \backslash \xi_{end}$	0.15	0.13	0.11	0.09	0.07	0.05	0.03	0.01
0.15	133	104	95	93	99	80	59	77
0.13	-	118	108	78	78	86	38	52
0.11	-	-	83	82	105	78	54	42
0.09	-	-	-	75	93	75	91	38
0.07	-	-	-	-	83	53	64	34
0.05	-	-	-	-	-	29	31	38
0.03	-	-	-	-	-	-	58	77
0.01	-	-	-	-	-	-	-	83

Table 4.9: Pair of (RP_{start}, RP_{end}) for B-SLRPSO (F2)

$RP_{start} \backslash RP_{end}$	0.75	0.8	0.85	0.9	0.95	0.99
0.75	21	20	18	17	11	6
0.8	-	19	16	14	9	5
0.85	-	-	15	13	10	3
0.9	-	-	-	12	8	4
0.95	-	-	-	-	7	1
0.99	-	-	-	-	-	2

Table 4.10: Pair of (RP_{start}, RP_{end}) for B-SLRPSO (F4)

$RP_{start} \backslash RP_{end}$	0.75	0.8	0.85	0.9	0.95	0.99
0.75	21	19	18	16	11	5
0.8	-	20	17	15	10	6
0.85	-	-	14	13	8	3
0.9	-	-	-	12	9	4
0.95	-	-	-	-	7	2
0.99	-	-	-	-	-	1

$(0.95, 0.99)$ for the following experiments.

Table 4.14 - Table 4.17 show the rank of the pairs of (RP_{start}, RP_{end}) of B-FLRPSO for F2, F4, F6 and F9. Table 4.18 is the summation of those ranks. The pairs $\{(0.90, 0.99)$, $(0.95, 0.99)$, $(0.99, 0.99)$ $\}$ seem to be better choices than others. We will use the pair $(0.95, 0.99)$ for the following experiments.

Table 4.11: Pair of (RP_{start}, RP_{end}) for B-SLRPSO (F6)

$RP_{start} \backslash RP_{end}$	0.75	0.8	0.85	0.9	0.95	0.99
0.75	20	19	13	6	10	11
0.8	-	21	14	7	2	16
0.85	-	-	12	5	3	17
0.9	-	-	-	9	18	4
0.95	-	-	-	-	15	1
0.99	-	-	-	-	-	8

Table 4.12: Pair of (RP_{start}, RP_{end}) for B-SLRPSO (F9)

$RP_{start} \backslash RP_{end}$	0.75	0.8	0.85	0.9	0.95	0.99
0.75	18	19	13	4	2	7
0.8	-	21	6	5	11	14
0.85	-	-	12	3	15	1
0.9	-	-	-	17	8	9
0.95	-	-	-	-	16	10
0.99	-	-	-	-	-	20

Table 4.13: Pair of (RP_{start}, RP_{end}) for B-SLRPSO (Sum)

$RP_{start} \backslash RP_{end}$	0.75	0.8	0.85	0.9	0.95	0.99
0.75	80	77	62	43	34	29
0.8	-	81	53	41	32	41
0.85	-	-	53	34	36	24
0.9	-	-	-	50	43	21
0.95	-	-	-	-	45	14
0.99	-	-	-	-	-	31

Experimental Design

We start with the main experiments in this section. We compare the following eight methods in the experiments of benchmark functions:

1. **OBPSO** in Section 2.5.3:
   We follow the parameter specification in Mirjalili and Lewis [108]. $(c_1, c_2) = (2, 2)$. Originally ω was set to 1 in Kennedy and Eberhart [72], but following [108] we adaptively adjust ω so that ω starts with 0.9 at the first iteration and lineally decreases to 0.4 at the last iteration. The maximum of the velocity is set to 6.

Table 4.14: Pair of (RP_{start}, RP_{end}) for B-FLRPSO (F2)

$RP_{start} \backslash RP_{end}$	0.75	0.8	0.85	0.9	0.95	0.99
0.75	21	20	19	17	11	6
0.8	-	18	16	14	10	5
0.85	-	-	15	13	9	4
0.9	-	-	-	12	8	3
0.95	-	-	-	-	7	2
0.99	-	-	-	-	-	1

Table 4.15: Pair of (RP_{start}, RP_{end}) for B-FLRPSO (F4)

$RP_{start} \backslash RP_{end}$	0.75	0.8	0.85	0.9	0.95	0.99
0.75	21	20	19	17	11	4
0.8	-	18	16	14	10	5
0.85	-	-	15	13	9	6
0.9	-	-	-	12	8	3
0.95	-	-	-	-	7	2
0.99	-	-	-	-	-	1

Table 4.16: Pair of (RP_{start}, RP_{end}) for B-FLRPSO (F6)

$RP_{start} \backslash RP_{end}$	0.75	0.8	0.85	0.9	0.95	0.99
0.75	21	20	19	15	12	7
0.8	-	18	17	13	8	4
0.85	-	-	16	14	10	5
0.9	-	-	-	11	9	2
0.95	-	-	-	-	6	3
0.99	-	-	-	-	-	1

2. **VBPSO** in Section 2.5.3:
   The parameter specification is the same as OBPSO.

3. **KBPSO** in Section 2.5.3:
   We follow the parameter specification in (Khanesar et al. [75], Khanesar [74]). $(\omega, c_1, c_2) = (0.5, 1, 1)$ and the maximum of the velocity is set to 4.

4. **PBPSO**: We follow the parameter specifications in Zhen et al. [181]. We set $(c_1, c_2) = (2, 2)$ in (2.34), the maximum of the velocity to 4 and P_m to 0.08.

Table 4.17: Pair of (RP_{start}, RP_{end}) for B-FLRPSO (F9)

$RP_{start} \backslash RP_{end}$	0.75	0.8	0.85	0.9	0.95	0.99
0.75	19	18	21	16	10	6
0.8	-	20	17	14	11	5
0.85	-	-	15	13	9	4
0.9	-	-	-	12	8	3
0.95	-	-	-	-	7	2
0.99	-	-	-	-	-	1

Table 4.18: Pair of (RP_{start}, RP_{end}) for B-FLRPSO (Sum)

$RP_{start} \backslash RP_{end}$	0.75	0.8	0.85	0.9	0.95	0.99
0.75	82	78	78	65	44	23
0.8	-	74	66	55	39	19
0.85	-	-	61	53	37	19
0.9	-	-	-	47	33	11
0.95	-	-	-	-	27	9
0.99	-	-	-	-	-	4

5. **WBPSO**: We follow the parameter specifications in Wang et al. [181], [180]. We set $(c_1, c_2) = (2, 2)$ in (2.34), the maximum of the velocity to 4 and P_m to 0.08.

6. **B-SPSO** in Section 4.3.3:
   From the preliminary experiments in Section 4.4.1, we set $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$, which is originally proposed in [41]. We also set the parameter ξ so that ξ is 0.08 at the first iteration and lineally decreases to 0.01 at the last iteration as discussed in the preliminary experiments above.

7. **B-SLRPSO** in Section 4.3.4:
   $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$ As discussed in the preliminary experiments, we set the ratio of preservation (RP) so that RP starts with 0.95 at the first iteration and lineally decreases to 0.99 at the last iteration.

8. **B-FLRPSO** in Section 4.3.4:
   As discussed in the preliminary experiments, we set $(\omega, c_1, c_2, c_3) = (0.5, 1.3, 1.3, 1.3)$. We set the ratio of preservation (RP) so that RP starts with 0.95 at the first iteration and lineally decreases to 0.99 at the last iteration.

In the experiments we use four basic functions (Sphere, Rosenbrock, Griewangk

and Rastrigin) and ten CEC 2005 benchmark functions in Suganthan et al. [150] as shown in Table 4.1. (The matlab codes are also provided in [150].) The global optimal values of those functions are all set to zero. Therefore the values which are closer to zero are better. Experiments are repeated 50 times for each function.

In this experiments, each real value in the input vector of the benchmark functions is represented by 20 bits of Gray code which we explain in Section 2.9. The dimension of continuous solutions are set to be 3, 6 and 9 and so the dimension of the binary code is 60, 120 and 180, respectively.

We conduct three experiments. In the first experiment we set the size of population and the number of iteration are both set to $100\times$ (dimension of real solution $/3$) for the basic or unimodal functions (S1, S2, S3, S4, F1, F2, F4, F5). We set the number of iteration to 500 and the size of population to $300\times$ (dimension of real solution $/3$) for the multi-modal benchmark functions (F6, F9, F10, F11, F12, F14).

In the second experiment we repeat the first experiment with larger sizes of population and iteration. We set the number of iteration to 600 and the size of population to $200\times$ (dimension of real solution $/3$) for the basic or unimodal functions (S1, S2, S3, S4, F1, F2, F4, F5), and set the number of iteration to 1000 and the size of population to $600\times$ (dimension of real solution $/3$) for the multi-modal benchmark functions (F6, F9, F10, F11, F12, F14).

In the third experiment we examine the performance of probability-based binary PSO methods in Section 4.3.1. We compare WBPSO, PBPSO, B-SPSO, B-SLRPSO and B-FLRPSO using the same setting in the first experiment.

Results

Tables 4.19, 4.20 and 4.21 report the results of optimal solutions on the 14 benchmark functions. For each function there are three rows and each row shows the results of the different dimension 3, 6 and 9 of real solution , which correspond to 60, 120 and 180 in the binary dimension as explained above. In each row we emphasize the best result in boldface. In Tables 4.19 and Table 4.20, we conduct the two-tailed t-test to compare the performance of KBPSO with B-FLRPSO. If the results of two methods are significantly different (p-values < 0.05), (*) is shown beside the number in the column of the better method.

In Tables 4.21, we conduct the two-tailed t-test to compare the performance of WBPSO with B-FLRPSO. If the results of two methods are significantly different (p-values < 0.05), (*) is shown beside the number in the column of the better method. We

also conduct the two-tailed t-test to compare the performance of PBPSO with B-SPSO. If the results of two methods are significantly different (p-values < 0.05), ($^{(*)}$) is shown beside the number in the column of the better method.

Table 4.19: Experiments with Benchmark Functions for Binary PSO

Func.	Dim.	**OBPSO**	**VBPSO**	**KBPSO**	**PBPSO**	**B-SPSO**	**B-SLRPSO**	**B-FLRPSO**
S1	3	81.2 ± 36.9	0.00394 ± 0.00899	$4.28e-05^{*}$ ± 6.15e-05	1.2e-06 ± 1.59e-06	**6.82e-09** ± 1.67e-24	**6.82e-09** ± 1.67e-24	0.000131 ± 0.000133
	6	508 ± 106	0.0117 ± 0.0149	0.0205 ± 0.0201	0.00315 ± 0.00202	2.16e-08 ± 2.87e-08	**1.4e-08** ± 2.57e-09	0.0108^{*} ± 0.00591
	9	1.04e+03 ± 144	0.0212 ± 0.0209	0.353 ± 0.205	0.135 ± 0.0613	2.28e-06 ± 1.89e-06	**3.76e-07** ± 1.06e-06	0.135^{*} ± 0.051
S2	3	2.57e+04 ± 2.18e+04	7.71 ± 10.6	7.04 ± 10.5	8.19 ± 11.1	6.11 ± 8.66	9.51 ± 11.4	**0.214*** ± 0.262
	6	3.1e+06 ± 1.76e+06	15.5 ± 17	16.5 ± 15.3	12 ± 12.3	8.04 ± 9.62	11.4 ± 12.1	**6.34*** ± 3.97
	9	1.44e+07 ± 3.58e+06	41.1 ± 37.9	126 ± 66.4	81.7 ± 40.7	**9.39** ± 11.1	11.6 ± 11.2	48.8^{*} ± 18.6
S3	3	0.217 ± 0.0456	0.0236 ± 0.0205	0.0162 ± 0.016	0.0209 ± 0.0175	0.024 ± 0.0212	0.0404 ± 0.0351	**0.00855*** ± 0.0052
	6	0.728 ± 0.105	0.0687 ± 0.0406	0.0609 ± 0.0294	0.0518 ± 0.0326	0.0371 ± 0.0288	0.0817 ± 0.0511	**0.0328*** ± 0.00949
	9	1.16 ± 0.0735	0.105 ± 0.043	0.112 ± 0.034	0.0995 ± 0.035	0.0704 ± 0.0334	0.0888 ± 0.0494	**0.0587*** ± 0.0145
S4	3	105 ± 38.5	2.02 ± 0.97	1.92 ± 0.683	1.72 ± 0.958	1.57 ± 0.831	1.95 ± 0.942	**0.762*** ± 0.455
	6	542 ± 144	4.63 ± 1.92	4.75 ± 1.64	3.96 ± 1.06	**3.34** ± 1.15	3.68 ± 1.16	3.53^{*} ± 0.77
	9	1.21e+03 ± 134	7.56 ± 2.32	16.5 ± 5.02	12.1 ± 2.59	5.73 ± 1.71	**5.55** ± 1.35	10.7^{*} ± 1.73
F1	3	176 ± 127	0.0046 ± 0.0114	0.000111 ± 0.000154	3.11e-06 ± 5.94e-06	**4.21e-09** ± 2.51e-24	**4.21e-09** ± 2.51e-24	0.000131 ± 0.000148
	6	2.79e+03 ± 1.12e+03	0.0184 ± 0.0235	0.041 ± 0.0294	0.0076 ± 0.00612	1.55e-08 ± 1.7e-08	**7.65e-09** ± 5.32e-09	0.018^{*} ± 0.0123
	9	7.82e+03 ± 1.74e+03	0.0557 ± 0.0656	1.05 ± 0.774	0.365 ± 0.164	5.46e-06 ± 6.16e-06	**6.5e-07** ± 1.44e-06	0.272^{*} ± 0.12
F2	3	205 ± 143	0.043 ± 0.0833	0.00278^{*} ± 0.0059	0.000104 ± 0.000185	1.48e-07 ± 2.43e-07	**9.43e-08** ± 1.62e-07	0.0143 ± 0.0222
	6	2.13e+03 ± 785	1.32 ± 1.72	0.729^{*} ± 0.963	0.11 ± 0.174	0.000679 ± 0.0011	**0.000206** ± 0.000342	2.92 ± 1.92
	9	7.78e+03 ± 1.95e+03	8.06 ± 9.24	13.7^{*} ± 10.7	5.1 ± 2.64	0.0544 ± 0.0714	**0.0162** ± 0.0164	27.2 ± 11.4
F4	3	262 ± 150	0.0435 ± 0.0935	0.00957^{*} ± 0.0378	0.000116 ± 0.000207	**1.16e-05** ± 4.38e-05	1.35e-05 ± 4.6e-05	0.0286 ± 0.0393
	6	2.88e+03 ± 931	2.19 ± 4.18	1.75^{*} ± 1.68	0.262 ± 0.229	**0.00923** ± 0.0255	0.00955 ± 0.0172	2.93 ± 2.11
	9	9.26e+03 ± 2.32e+03	17.2 ± 23.8	22.3^{*} ± 18	8.41 ± 7.26	**0.385** ± 0.67	0.572 ± 0.996	48.8 ± 22.1
F5	3	1.14e+03 ± 366	26.3 ± 24.9	12.8^{*} ± 16.5	11.6 ± 38.7	**10.5** ± 27.6	51.8 ± 68.2	32.9 ± 25.7
	6	8.94e+03 ± 1.65e+03	63.3 ± 72.5	78.5 ± 38	32 ± 15.9	**0.99** ± 1.12	1.41 ± 2.64	84.8 ± 27.8
	9	1.22e+04 ± 1.38e+03	192 ± 170	315 ± 126	185 ± 88.7	**18.4** ± 20.4	55 ± 143	262^{*} ± 57.8
F6	3	2.05e+04 ± 2.38e+04	4.17 ± 9.95	2.81 ± 7.26	9.1 ± 12.8	11.3 ± 16.4	12.1 ± 17.6	**0.00752*** ± 0.0142
	6	1.61e+07 ± 8.35e+06	24.8 ± 57.9	16.3 ± 41.5	9.79 ± 16.4	10 ± 22.5	12.3 ± 33.2	**0.606*** ± 0.489
	9	2.55e+08 ± 1.36e+08	40.9 ± 73	104 ± 84.2	74.6 ± 56.9	16.5 ± 34.1	**13.1** ± 26.9	15.9^{*} ± 4.86
F9	3	3.35 ± 0.971	0.0199 ± 0.141	1.43e-08 ± 6.93e-08	0.0995 ± 0.302	0.0796 ± 0.273	0.119 ± 0.327	**3.87e-09** ± 3.34e-24
	6	22.9 ± 3.88	0.618 ± 0.692	0.626 ± 0.744	0.399 ± 0.568	0.478 ± 0.611	0.418 ± 0.639	**0.000763*** ± 0.000805
	9	58.3 ± 6.86	2.28 ± 1.3	2.55 ± 1.18	2.2 ± 1.1	1.51 ± 1.16	1.45 ± 1.03	**1.19*** ± 0.624
F10	3	4.08 ± 1.39	0.279 ± 0.494	0.517 ± 0.674	0.836 ± 0.838	0.895 ± 0.859	1.35 ± 1.02	**1.12e-06*** ± 7.84e-06
	6	34.7 ± 5.6	4.29 ± 2.04	5.77 ± 2.52	5.76 ± 2.93	6.05 ± 2.78	8.44 ± 4.58	**1.25*** ± 0.634
	9	85.2 ± 9.05	12.5 ± 4.91	12.5 ± 5.39	13.8 ± 6.54	11.9 ± 5.16	23.9 ± 9.83	**4.95*** ± 1.21
F11	3	0.654 ± 0.115	0.0387 ± 0.056	0.0542 ± 0.0945	0.106 ± 0.183	0.0726 ± 0.125	0.265 ± 0.358	**0.0104*** ± 0.00785
	6	3.44 ± 0.395	0.724 ± 0.59	1.19 ± 0.767	1.24 ± 0.919	1.05 ± 0.784	1.66 ± 0.864	**0.377*** ± 0.126
	9	6.94 ± 0.5	2.66 ± 1.23	2.92 ± 1.08	2.54 ± 1.22	2.13 ± 1.14	3.86 ± 1.18	**1.77*** ± 0.306
F12	3	46.4 ± 40	0.0433 ± 0.106	0.035 ± 0.0878	1.6 ± 6.26	1.05 ± 5.17	5.22 ± 10.6	**0.00189*** ± 0.00191
	6	4.53e+03 ± 1.98e+03	51 ± 70.4	81.5 ± 72.6	64.2 ± 50.1	20.1 ± 30.1	24 ± 48.6	**8.37*** ± 5.07
	9	1.73e+04 ± 5.05e+03	988 ± 901	2.47e+03 ± 990	1.7e+03 ± 1.04e+03	**416** ± 559	506 ± 711	596^{*} ± 239
F14	3	0.817 ± 0.16	0.0721 ± 0.141	0.0391 ± 0.0311	0.191 ± 0.341	0.137 ± 0.273	0.279 ± 0.372	**0.0259*** ± 0.0186
	6	1.88 ± 0.111	1.19 ± 0.43	1.26 ± 0.366	1.37 ± 0.357	1.3 ± 0.48	1.5 ± 0.333	**1.04*** ± 0.208
	9	3.26 ± 0.153	**2.43** ± 0.517	2.63 ± 0.315	2.65 ± 0.368	2.44 ± 0.396	2.79 ± 0.378	2.47^{*} ± 0.191

Table 4.19 and Table 4.20 indicates that B-SPSO and B-SLRPSO work best for the unimodal functions (S1,F1,F2,F4,F5) and B-FLRPSO works best for the multi-modal functions (S2,S3,S4,F6-F14). Among the "velocity as probability" methods, VBPSO and KBPSO work better than OBPSO. Among the "position as probability" methods, the proposed B-SPSO and PBPSO show the similar performance. We also notice that the performance of VBPSO and KBPSO is comparable to that of B-FLRPSO for every function. Those results are attributed to the similarity of the convergence speed

Table 4.20: Experiments with Benchmark Functions for Binary PSO - Larger Size

Func.	Dim.	OBPSO	VBPSO	KBPSO	PBPSO	B-SPSO	B-SLRPSO	B-FLRPSO
S1	3	50.5 ± 26.6	**6.82e-09** ± 1.67e-24	**6.82e-09** ± 1.67e-24	**6.82e-09** ± 1.67e-24	**6.82e-09** ± 1.67e-24	**6.82e-09** ± 1.67e-24	**6.82e-09** ± 1.67e-24
	6	397 ± 89.7	3.08e-06 ± 2.12e-05	0.000149 ± 0.000138	3.69e-05 ± 2.63e-05	**1.36e-08** ± 1e-23	**1.36e-08** ± 1e-23	$1.15e-05^{*}$ ± 6.63e-06
	9	956 ± 124	0.000271 ± 0.00135	0.0332 ± 0.0242	0.0163 ± 0.00861	**2.05e-08** ± 3.34e-24	**2.05e-08** ± 3.34e-24	0.0059^{*} ± 0.00232
S2	3	6.87e+03 ± 6.29e+03	1.26 ± 2.91	1.48 ± 3.78	6.11 ± 8.71	4.34 ± 6.72	5.63 ± 7.92	**0.00999*** ± 0.0299
	6	1.74e+06 ± 9.01e+05	5.52 ± 7.44	8.14 ± 10.1	6.69 ± 8.11	7.23 ± 9.68	12 ± 13.1	**0.118*** ± 0.141
	9	1.15e+07 ± 3.72e+06	8.45 ± 11.3	40.5 ± 26.2	34.5 ± 26.5	8.92 ± 10.2	13.4 ± 10.9	**5.74*** ± 2.68
S3	3	0.142 ± 0.0295	0.00577 ± 0.0043	0.00666 ± 0.00339	0.00991 ± 0.0103	0.00937 ± 0.0106	0.0172 ± 0.0162	**0.00148*** ± 0.00299
	6	0.612 ± 0.0848	0.0291 ± 0.0173	0.028 ± 0.0137	0.0228 ± 0.0162	0.0238 ± 0.0156	0.0337 ± 0.0215	**0.0102*** ± 0.00496
	9	1.1 ± 0.07	0.0732 ± 0.0256	0.0857 ± 0.0349	0.0644 ± 0.0265	0.0493 ± 0.0188	0.0542 ± 0.0262	**0.0257*** ± 0.00955
S4	3	81.7 ± 23.2	0.756 ± 0.683	0.817 ± 0.626	0.904 ± 0.673	1.05 ± 0.736	1.49 ± 0.759	**9.29e-06*** ± 5.41e-05
	6	466 ± 85.1	3.28 ± 1.24	3.39 ± 1.25	2.88 ± 1.09	3.16 ± 1.15	3.28 ± 1.4	**1.44*** ± 0.552
	9	1.02e+03 ± 151	6.41 ± 1.76	7.9 ± 2.15	6.87 ± 1.22	5.13 ± 1.5	5.29 ± 1.52	**4.66*** ± 0.788
F1	3	77.4 ± 48	**4.21e-09** ± 2.51e-24	**4.21e-09** ± 2.51e-24	**4.21e-09** ± 2.51e-24	**4.21e-09** ± 2.51e-24	**4.21e-09** ± 2.51e-24	**4.21e-09** ± 2.51e-24
	6	2.18e+03 ± 863	1.34e-06 ± 4.31e-06	0.00046 ± 0.00046	0.000124 ± 7.79e-05	**6.9e-09** ± 5.01e-24	**6.9e-09** ± 5.01e-24	$1.52e-05^{*}$ ± 7.76e-06
	9	6.42e+03 ± 1.49e+03	0.000167 ± 0.000494	0.104 ± 0.0634	0.054 ± 0.0282	**7.41e-09** ± 5.01e-24	**7.41e-09** ± 5.01e-24	0.013^{*} ± 0.0061
F2	3	93.9 ± 49.5	1.48e-08 ± 1.07e-08	3.24e-08 ± 7.09e-08	**1.18e-08** ± 1.02e-10	**1.18e-08** ± 1.67e-24	**1.18e-08** ± 1.67e-24	$1.22e-08^{*}$ ± 2.79e-09
	6	1.66e+03 ± 515	0.00224 ± 0.0108	0.0122 ± 0.0148	0.00178 ± 0.00148	2.68e-08 ± 2.28e-08	**2.19e-08** ± 3.04e-08	0.0125 ± 0.00846
	9	6.45e+03 ± 1.73e+03	0.141 ± 0.204	1.79^{*} ± 1.4	0.695 ± 0.344	0.000161 ± 0.000257	**2.19e-05** ± 3.51e-05	3.03 ± 1.16
F4	3	108 ± 63.9	6.8e-08 ± 3.89e-07	2.52e-08 ± 3.94e-08	1.2e-08 ± 3.09e-10	**1.18e-08** ± 1.02e-10	**1.18e-08** ± 4.87e-13	2.03e-08 ± 4.62e-08
	6	2.11e+03 ± 914	0.0132 ± 0.0581	0.0198^{*} ± 0.0202	0.00402 ± 0.0053	1.43e-06 ± 3.44e-06	**7.11e-07** ± 2.07e-06	0.0293 ± 0.0238
	9	7.93e+03 ± 1.9e+03	0.843 ± 1.32	3.28^{*} ± 2.47	1.11 ± 0.953	**0.00358** ± 0.0056	0.00527 ± 0.0113	6.29 ± 3.32
F5	3	940 ± 242	0.0351 ± 0.16	0.0365^{*} ± 0.0468	0.00751 ± 0.0151	**0.000145** ± 0.00103	0.000912 ± 0.00284	0.345 ± 0.401
	6	7.79e+03 ± 1.36e+03	1.72 ± 4.91	7.69 ± 3.54	3.54 ± 1.26	**0.000871** ± 0.00126	0.0024 ± 0.00373	4.59^{*} ± 1.46
	9	1.17e+04 ± 1.15e+03	32.3 ± 48.7	115 ± 57.1	66.1 ± 23.5	**1.99** ± 2.66	2.56 ± 3.11	65.6^{*} ± 13.4
F6	3	9.67e+03 ± 1.09e+04	0.864 ± 3.6	0.652 ± 2.38	1.29 ± 4.89	0.115 ± 0.369	4.36 ± 8.2	**0.000912** ± 0.00215
	6	8.7e+06 ± 3.93e+06	3.85 ± 6.94	8.63 ± 21.3	5.34 ± 7.24	4.07 ± 6.39	4.1 ± 7.29	**0.0439*** ± 0.052
	9	1.82e+08 ± 7.21e+07	19.6 ± 32.3	51.3 ± 79	26.8 ± 39.9	7.06 ± 7.97	7.65 ± 7.81	**3.9*** ± 1.32
F9	3	2.53 ± 0.754	**3.87e-09** ± 3.34e-24	**3.87e-09** ± 3.34e-24	**3.87e-09** ± 3.34e-24	**3.87e-09** ± 3.34e-24	0.0597 ± 0.239	**3.87e-09** ± 3.34e-24
	6	20.2 ± 3.37	0.179 ± 0.386	0.179 ± 0.386	0.0796 ± 0.339	0.0995 ± 0.302	0.338 ± 0.476	**9.07e-07*** ± 6.34e-07
	9	50.2 ± 6.79	1.39 ± 1.05	1.54 ± 1.01	1.21 ± 1	0.776 ± 0.76	1.05 ± 0.886	**0.0453*** ± 0.143
F10	3	2.35 ± 0.739	0.139 ± 0.349	0.179 ± 0.386	0.796 ± 0.778	0.716 ± 0.78	0.836 ± 0.907	**5.69e-09*** ± 2.51e-24
	6	30.4 ± 5.07	3.19 ± 1.77	4.48 ± 2.07	4.39 ± 2.27	3.8 ± 1.88	6.94 ± 3.56	**0.896*** ± 0.461
	9	80.5 ± 9.77	9.77 ± 3.72	11.7 ± 5.75	11.9 ± 5.09	11.6 ± 4.4	17.2 ± 6.76	**2.88*** ± 0.713
F11	3	0.674 ± 0.125	0.0272 ± 0.0741	0.049 ± 0.119	0.0487 ± 0.138	0.0332 ± 0.101	0.104 ± 0.204	**0.00128*** ± 0.000302
	6	2.77 ± 0.346	0.538 ± 0.485	0.916 ± 0.638	0.829 ± 0.689	0.792 ± 0.775	1.3 ± 0.889	**0.149*** ± 0.056
	9	6.41 ± 0.411	2.3 ± 1.02	2.46 ± 1.06	2.28 ± 0.979	1.56 ± 1.13	3.51 ± 1.14	**0.999*** ± 0.222
F12	3	15.8 ± 11	0.000214 ± 0.000456	0.00116 ± 0.00493	0.000485 ± 0.00222	0.522 ± 3.69	3.13 ± 8.57	**8.79e-05** ± 8.21e-05
	6	3.01e+03 ± 1.23e+03	23.1 ± 33.3	35.4 ± 47.2	29.6 ± 37.5	9.85 ± 21.2	9.72 ± 17.9	**1.04*** ± 0.735
	9	1.16e+04 ± 3.29e+03	360 ± 517	999 ± 620	866 ± 734	295 ± 598	309 ± 635	**130*** ± 46.5
F14	3	0.637 ± 0.165	0.0257 ± 0.00471	0.0292 ± 0.0187	0.0437 ± 0.0422	0.0295 ± 0.0209	0.145 ± 0.259	**0.0185*** ± 0.00491
	6	1.66 ± 0.158	0.944 ± 0.442	1.08 ± 0.441	1.24 ± 0.38	1.18 ± 0.372	1.45 ± 0.297	**0.59*** ± 0.216
	9	3.12 ± 0.132	2.18 ± 0.434	2.32 ± 0.408	2.34 ± 0.448	2.16 ± 0.469	2.54 ± 0.386	**1.9*** ± 0.215

among those methods as shown in Figure 4.16 and Figure 4.17. However, in Table 4.19 B-FLRPSO is significantly better ($p < 0.05$) than KBPSO for 31 out of 42 rows and significantly worse than KBPSO for 7 rows. In Table 4.20 B-FLRPSO is significantly better ($p < 0.05$) than KBPSO for 31 out of 42 rows and significantly worse than KBPSO for 4 rows. Overall the "position as probability" methods outperform the "velocity as probability" methods.

In Table 4.21, we examine the performance of probability-based binary PSO methods in Section 4.3.1. We compared the performance of WBPSO and B-FLRPSO using t test. B-FLRPSO is significantly better ($p < 0.05$) than WBPSO for all 42 rows Therefore, we verified that our approach in which pbest are evaluated only at the deiscrete points

Table 4.21: Comparison of Probability-based Binary PSO Methods

Func.	Dim.	**WBPSO**	**PBPSO**	**B-SPSO**	**B-SLRPSO**	**B-FLRPSO**
S1	3	1.41 ± 2.96	1.76e-06 ± 4.59e-06	**6.82e-09**$^{(*)}$ ± 1.67e-24	**6.82e-09** ± 1.67e-24	0.000131^{*} ± 0.000133
	6	54.9 ± 49.6	0.00301 ± 0.0027	$2.16e-08^{(*)}$ ± 2.87e-08	**1.4e-08** ± 2.57e-09	0.0108^{*} ± 0.00591
	9	218 ± 160	0.109 ± 0.0465	$2.28e-06^{(*)}$ ± 1.89e-06	**3.76e-07** ± 1.06e-06	0.135^{*} ± 0.051
S2	3	216 ± 1.2e+03	9.13 ± 10.9	6.11 ± 8.66	9.51 ± 11.4	**0.214** ± 0.262
	6	4.51e+04 ± 8e+04	14.4 ± 13.5	$8.04^{(*)}$ ± 9.62	11.4 ± 12.1	**6.34*** ± 3.97
	9	7.89e+05 ± 1.63e+06	73.3 ± 43.6	**9.39**$^{(*)}$ ± 11.1	11.6 ± 11.2	48.8^{*} ± 18.6
S3	3	0.0596 ± 0.0419	0.0209 ± 0.0192	0.024 ± 0.0212	0.0404 ± 0.0351	**0.00855*** ± 0.0052
	6	0.286 ± 0.112	0.0449 ± 0.0236	0.0371 ± 0.0288	0.0817 ± 0.0511	**0.0328*** ± 0.00949
	9	0.662 ± 0.187	0.1 ± 0.0373	$0.0704^{(*)}$ ± 0.0334	0.0888 ± 0.0494	**0.0587*** ± 0.0145
S4	3	10.2 ± 8.64	1.65 ± 0.817	1.57 ± 0.831	1.95 ± 0.942	**0.762*** ± 0.455
	6	96.4 ± 57.5	3.97 ± 1.09	**3.34**$^{(*)}$ ± 1.15	3.68 ± 1.16	3.53^{*} ± 0.77
	9	245 ± 115	12.2 ± 2.72	$5.73^{(*)}$ ± 1.71	**5.55** ± 1.35	10.7^{*} ± 1.73
F1	3	4.28 ± 9.45	2.55e-06 ± 4.44e-06	**4.21e-09**$^{(*)}$ ± 2.51e-24	**4.21e-09** ± 2.51e-24	0.000131^{*} ± 0.000148
	6	174 ± 214	0.00774 ± 0.00495	$1.55e-08^{(*)}$ ± 1.7e-08	**7.65e-09** ± 5.32e-09	0.018^{*} ± 0.0123
	9	483 ± 412	0.369 ± 0.158	$5.46e-06^{(*)}$ ± 6.16e-06	**6.5e-07** ± 1.44e-06	0.272^{*} ± 0.12
F2	3	4.8 ± 10.6	0.000117 ± 0.00025	$1.48e-07^{(*)}$ ± 2.43e-07	**9.43e-08** ± 1.62e-07	0.0143^{*} ± 0.0222
	6	199 ± 267	0.149 ± 0.118	$0.000679^{(*)}$ ± 0.0011	**0.000206** ± 0.000342	2.92^{*} ± 1.92
	9	1.07e+03 ± 1e+03	4.44 ± 2.46	$0.0544^{(*)}$ ± 0.0714	**0.0162** ± 0.0164	27.2^{*} ± 11.4
F4	3	21.4 ± 63.4	0.000172 ± 0.000381	**1.16e-05**$^{(*)}$ ± 4.38e-05	1.35e-05 ± 4.6e-05	0.0286^{*} ± 0.0393
	6	257 ± 220	0.252 ± 0.322	**0.00923**$^{(*)}$ ± 0.0255	0.00955 ± 0.0172	2.93^{*} ± 2.11
	9	1.32e+03 ± 963	8.92 ± 6.11	**0.385**$^{(*)}$ ± 0.67	0.572 ± 0.996	48.8^{*} ± 22.1
F5	3	258 ± 234	**4.59** ± 6.29	10.5 ± 27.6	51.8 ± 68.2	32.9^{*} ± 25.7
	6	2.05e+03 ± 983	27 ± 11.9	**0.99**$^{(*)}$ ± 1.12	1.41 ± 2.64	84.8^{*} ± 27.8
	9	4.18e+03 ± 1.28e+03	170 ± 55.3	**18.4**$^{(*)}$ ± 20.4	55 ± 143	262^{*} ± 57.8
F6	3	18.1 ± 25.9	14.2 ± 19.8	11.3 ± 16.4	12.1 ± 17.6	**0.00752*** ± 0.0142
	6	3.59e+04 ± 6.35e+04	14.5 ± 32.4	10 ± 22.5	12.3 ± 33.2	**0.606*** ± 0.489
	9	5.15e+06 ± 2.39e+07	74.3 ± 62.1	$16.5^{(*)}$ ± 34.1	**13.1** ± 26.9	15.9 ± 4.86
F9	3	0.66 ± 0.637	0.0398 ± 0.197	0.0796 ± 0.273	0.119 ± 0.327	**3.87e-09*** ± 3.34e-24
	6	5.68 ± 2.75	0.43 ± 0.653	0.478 ± 0.611	0.418 ± 0.639	**0.000763*** ± 0.000805
	9	15.1 ± 4.96	1.93 ± 1.27	1.51 ± 1.16	1.45 ± 1.03	**1.19*** ± 0.624
F10	3	1.2 ± 0.961	0.876 ± 0.742	0.895 ± 0.859	1.35 ± 1.02	**1.12e-06*** ± 7.84e-06
	6	9.98 ± 3.46	5.59 ± 2.8	6.05 ± 2.78	8.44 ± 4.58	**1.25*** ± 0.634
	9	28.1 ± 6.96	11.9 ± 5.7	11.9 ± 5.16	23.9 ± 9.83	**4.95*** ± 1.21
F11	3	0.325 ± 0.256	0.102 ± 0.202	0.0726 ± 0.125	0.265 ± 0.358	**0.0104*** ± 0.00785
	6	2.38 ± 0.801	1.05 ± 0.736	1.05 ± 0.784	1.66 ± 0.864	**0.377*** ± 0.126
	9	6.07 ± 1.17	3.21 ± 1.07	$2.13^{(*)}$ ± 1.14	3.86 ± 1.18	**1.77*** ± 0.306
F12	3	4.79 ± 7.45	2.24 ± 7.19	1.05 ± 5.17	5.22 ± 10.6	**0.00189*** ± 0.00191
	6	611 ± 474	56.2 ± 74.8	$20.1^{(*)}$ ± 30.1	24 ± 48.6	**8.37*** ± 5.07
	9	5.84e+03 ± 2.69e+03	1.62e+03 ± 935	**416**$^{(*)}$ ± 559	506 ± 711	596^{*} ± 239
F14	3	0.273 ± 0.294	0.183 ± 0.333	0.137 ± 0.273	0.279 ± 0.372	**0.0259*** ± 0.0186
	6	1.6 ± 0.264	1.23 ± 0.406	1.3 ± 0.48	1.5 ± 0.333	**1.04*** ± 0.208
	9	2.98 ± 0.242	2.44 ± 0.363	**2.44** ± 0.396	2.79 ± 0.378	2.47^{*} ± 0.191

leads to better performance compared to WBPSO in which pbest can take any values in the space. We also compared the performance of PBPSO and B-SPSO using t test. B-SPSO is significantly better ($p < 0.05$) than PBPSO for 23 out of 42 rows in Table 4.21.

4.4.2 Experiments for Binary PSO using Knapsack Problems

In a Knapsack Problem, there are n items in total and each item j produces p_j profit and consumes m types of resources $\{r_{j1}, \ldots, r_{jm}\}$. Each resource has the capacity C_k, $k = 1, \ldots, m$. The task of the Knapsack Problem is to select a subset of items which produces the maximum profit under the condition that the consumption of each resource does not exceed the capacity. The problem is formulated as follows (Nguyen et al. [115]).

$$\begin{aligned} &\max_{\mathbf{x}} \sum_{j=1}^{n} p_j x_j \\ &\text{s.t. } r_{1k}x_1 +, \ldots, +r_{nk}x_n \leq C_k \quad k = 1, \ldots, m \\ &\quad x_j \in \{0, 1\} \quad j = 1, \ldots, n \end{aligned} \tag{4.26}$$

It is converted to the unconstrained problem using a large number ν.

$$\max_{\mathbf{x}} \sum_{j=1}^{n} p_j x_j + \nu \sum_{k=1}^{m} \min \left(C_k - \sum_{j=1}^{n} r_{jk} x_j, 0 \right)$$

We set $\nu = 1000$ in the experiments.

Experimental Design

We use the same setting as the first experiment in the previous section except that OBPSO is replaced with SBPSO in Section 2.5.3 whose parameter setting is specified as follows.

- **SBPSO**:
  We follow the parameter specification in Nguyen et al. [115], $(i_m, i_p, i_g) = (0.25, 0.25, 0.50)$ and maxLife $= 50$.

In the experiments we use the same knapsack problems in Nguyen et al. [115] as shown in Table 4.22. Those datasets are obtained from (Drake [39]).

We set the size of population to the number of item size in Table 4.22 and the number of iteration to 2000. Experiments are repeated 50 times for each dataset.

Results

In Table 4.23, the best results are emphasized in boldface. Note that in this table larger values are better. We compare the performance of SBPSO and B-FLRPSO using t test.

Table 4.22: Knapsack Datasets (Drake [39])

Dataset	Item Size	Resource Size
Pet7	50	5
Sento1	60	30
Sento2	60	30
Weish10	50	5
Weish15	60	5
Weish20	70	5
Weish25	80	5
Weish30	90	5
Gk01	100	15
Gk02	100	25
Gk03	150	25
Gk04	150	50
Weing7	105	2
Weing8	105	2

Table 4.23: Experiments of Knapsack Problems for Binary PSO

Problem	**VBPSO**	**KBPSO**	**SBPSO**	**PBPSO**	**B-SPSO**	**B-SLRPSO**	**B-FLRPSO**
Pet7	16454.6 $\pm$ 50.5	16467.9 $\pm$ 45.1	16443.8 $\pm$ 53.9	16475.1 $\pm$ 43.2	16464.7 $\pm$ 54.4	16398.1 $\pm$ 211.4	**16514.6*** $\pm$ 20.8
Sento1	7667.7 $\pm$ 88.3	7744.5 $\pm$ 29.3	7731.5 $\pm$ 44.4	7750.5 $\pm$ 21.4	7751.6 $\pm$ 22.3	7728.0 $\pm$ 46.4	**7765.7*** $\pm$ 7.8
Sento2	8679.1 $\pm$ 27.2	8698.2 $\pm$ 16.0	8701.8 $\pm$ 16.6	8708.3 $\pm$ 11.1	8700.7 $\pm$ 12.8	8691.5 $\pm$ 23.8	**8714.7*** $\pm$ 6.6
Weish10	6306.2 $\pm$ 50.0	6331.2 $\pm$ 18.3	6328.6 $\pm$ 20.7	6333.3 $\pm$ 16.7	6332.4 $\pm$ 16.4	6321.3 $\pm$ 25.5	**6338.9*** $\pm$ 0.2
Weish15	7448.9 $\pm$ 34.4	7473.0 $\pm$ 21.6	7479.2 $\pm$ 17.0	7482.2 $\pm$ 10.4	7476.8 $\pm$ 17.5	7469.3 $\pm$ 23.7	**7486.0*** $\pm$ 0.0
Weish20	9416.9 $\pm$ 39.2	9433.4 $\pm$ 26.0	9438.9 $\pm$ 17.1	9442.3 $\pm$ 11.6	9438.7 $\pm$ 13.3	9436.3 $\pm$ 18.3	**9448.2*** $\pm$ 4.1
Weish25	9886.8 $\pm$ 49.9	9900.2 $\pm$ 25.3	9921.5 $\pm$ 10.6	9915.9 $\pm$ 14.6	9925.6 $\pm$ 11.2	9915.3 $\pm$ 15.1	**9933.6*** $\pm$ 7.3
Weish30	11147.1 $\pm$ 35.7	11137.9 $\pm$ 27.8	11167.1 $\pm$ 14.7	11148.4 $\pm$ 22.1	11181.4 $\pm$ 12.1	11169.6 $\pm$ 14.9	**11189.2*** $\pm$ 3.9
Gk01	3704.9 $\pm$ 12.0	3689.4 $\pm$ 10.5	3699.8 $\pm$ 13.8	3696.4 $\pm$ 9.1	**3720.9** $\pm$ 10.8	3707.0 $\pm$ 13.4	3703.2 $\pm$ 9.0
Gk02	3893.3 $\pm$ 12.0	3871.5 $\pm$ 12.3	3887.6 $\pm$ 15.6	3882.6 $\pm$ 11.3	**3905.6** $\pm$ 9.8	3894.1 $\pm$ 14.8	3888.1 $\pm$ 8.2
Gk03	5553.3 $\pm$ 14.5	5517.7 $\pm$ 12.4	5538.7 $\pm$ 15.1	5525.8 $\pm$ 14.7	**5570.9** $\pm$ 14.5	5563.7 $\pm$ 14.2	5541.1 $\pm$ 9.9
Gk04	5662.5 $\pm$ 15.4	5630.2 $\pm$ 14.5	5455.8 $\pm$ 1030.5	5450.2 $\pm$ 1029.4	**5677.4** $\pm$ 14.0	5666.8 $\pm$ 17.3	5650.6 $\pm$ 7.6
Weing7	1092518.0 $\pm$ 2153.7	1091780.0 $\pm$ 1805.7	1093942.0 $\pm$ 1401.5	1092200.0 $\pm$ 1356.8	1094822.0 $\pm$ 896.3	1094198.0 $\pm$ 1334.2	**1095378.0*** $\pm$ 70.8
Weing8	613342.2 $\pm$ 11395.2	614102.8 $\pm$ 4247.0	619589.2 $\pm$ 3215.1	613952.8 $\pm$ 3647.9	622043.4 $\pm$ 2036.0	621860.8 $\pm$ 1731.5	**624358.2*** $\pm$ 2079.9

If the results between the two methods are significantly different (p-value < 0.05), (*) is shown beside the number in the column of the better result.

Table 4.23 indicates that the "position as probability" methods outperform the "velocity as probability" methods. (Note that in Table 4.23 results with larger values are better.) In Table 4.23, we compare the performance of SBPSO and B-FLRPSO using t

test. For 10 out of 14 problems, B-FLRPSO are significantly better than SBPSO.

4.4.3 Convergence Properties of Binary PSO Methods

In this section we examine the convergence properties of the binary PSO methods. Figure 4.16 shows how fast the points in the neighborhood of gbest with one Euclidean distance are explored by particles and Figure 4.17 shows how fast the average distances between pbest and gbest decrease. F1 dataset is used in the experiments with the size of population 50, the number of iteration 1000 and the dimension 3.

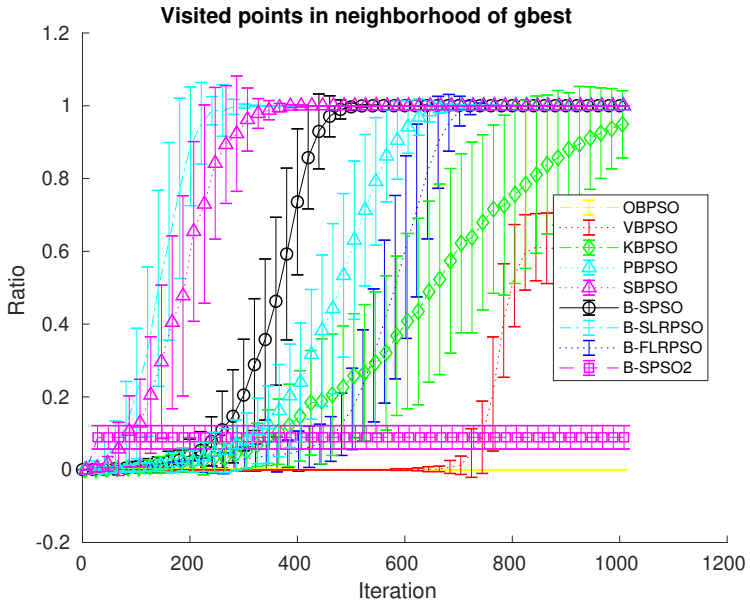


Figure 4.16: The ratio of visited points in the neighborhood of gbest with one Euclidean distance

For SBPSO, B-SLRPSO and B-SPSO the points in the neighborhood of gbest are quickly explored and the average distances between pbest and gbest decrease rapidly. The convergence speed of B-FLRPSO is relatively slow and is comparable to PBPSO, KBPSO and VBPSO. For OBPSO, the ratio of explored points in the neighborhood of gbest remains near zero and the average distance between pbest and gbest also remains unchanged as that of the initial iteration. We also show that if we set no perturbation ($\xi = 0$ in (4.17)) in B-SPSO which is labeled as B-SPSO2 in the figures, then it converges to the local optimum without exploring the neighborhood of gbest.

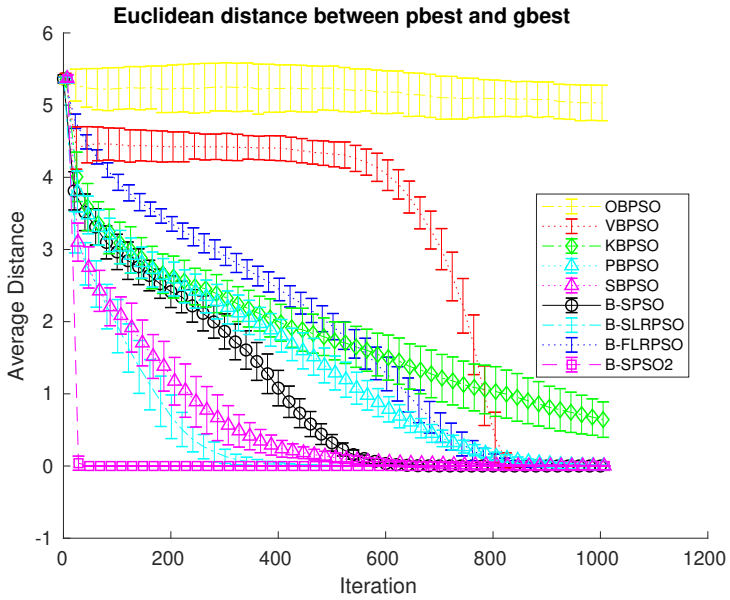


Figure 4.17: The average Euclidean distance between pbest and gbest

4.4.4 Experiments for Discrete PSO with Round Functions using Independent Job Scheduling Problems

Discrete PSO with round functions can be applied to the combinatorial optimization problems which seek for the optimal sequence of integers with repetition (which allows repetitions of the same integers in the sequence). As an example, we consider a scheduling problem of independent jobs in a heterogeneous computing (HC) environment such as grid computing, which is known as a NP-Hard problem (Ritchie and Levine [127], Garey and Johnson [50]). Suppose that there are n jobs and m machines and the estimated times E_{ij} for completing the j-th job on the i-th machine are stored in the ETC (Expected Time to Compute) matrix. This problem is modeled by a discrete PSO in which particles have n elements (n dimension) corresponding to the n jobs and each element takes a value which represents one of the m machines. The task is to allocate each job to a machine so as to minimize the following criteria (French [47]):

- Make-span: A make-span is a time to complete the latest job and computed as:

$$C_{max} = \max_i \left(\sum_j E_{ij} \right)$$

  where $\sum_j E_{ij}$ is the sum of E_{ij} for all jobs j allocated to machine i.

- Mean flow-time: A mean flow-time is the mean time to complete all jobs on each

machine and computed as:

$$\bar{F} = \text{mean}_i \left(\sum_j E_{ij} \right)$$

In this experiment, we solve the optimization problem of the combined criteria:

$$\min \lambda C_{max} + (1 - \lambda)\bar{F}$$

with $\lambda = 0.7$ (Izakian et al. [67]).

In this section we also start with the preliminary experiments to specify the perturbation parameters.

Preliminary Experiments

Specification of Perturbation Parameters: In this section we examine the optimal specification of perturbation parameter σ for D-SLRPSO and D-FLRPSO in (4.19) and (4.19) in Section 4.3.4. Note that D-SPSO in Section 4.3.3 does not have a perturbation parameter. Since we deal with the sequence of integers in this section, we use the ratio of preservation (RP) as in binary PSO to find the optimal settings of perturbation parameters. We start with the experiments by visualizing how the optimal values of RP change according to the different dimension of spaces and the different sizes of population. We conduct experiments with iteration $= 500$ and the number of jobs $= 128$ using "inconsistent and partially consistent" ETC matrices which are explained in the following section.

Figure 4.18 - Figure 4.21 are the contour maps of the optimal solutions for the input space of the ratio of preservation $\in \{0.60, 0.70, 0.80, 0.85, 0.90, 0.95, 0.99\}$ vs. the size of dimension (number of machines) $\in \{16, 32, 64, 128, 256\}$ and the ratio of preservation vs. the size of population $\in \{50, 100, 200, 500\}$ for D-SLRPSO and D-FLRPSO. The left figure is the one for an "inconsistent" ETC matrix and the right figure is the one for a "partially consistent" ETC matrix.

Next we determine the best combination of the ratio of preservation (RP). Based on Figure 4.18 - Figure 4.21, we set the range

$$RP \in \{0.80, 0.90, 0.92, 0.94, 0.96, 0.98\} \tag{4.27}$$

and decide the best pair of (RP_{start}, RP_{end}) from this range under the condition that $RP_{start} \leq RP_{end}$. We evaluate each pair using three ETC matrices; "consistent", "inconsistent" and "partially consistent". We set the dimension (the number of machines)

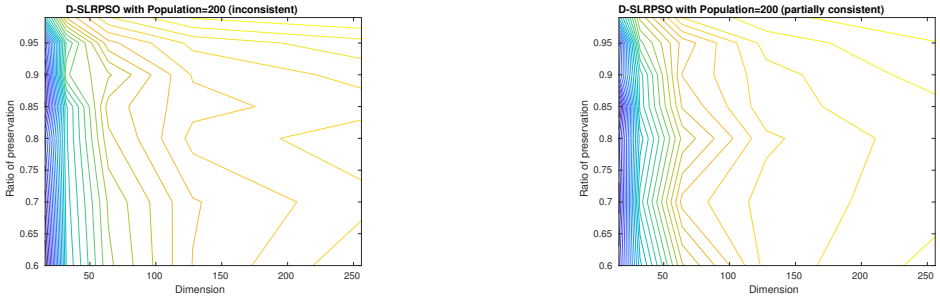


Figure 4.18: Ratio of Preservation vs. Dimension of Space (D-SLRPSO)

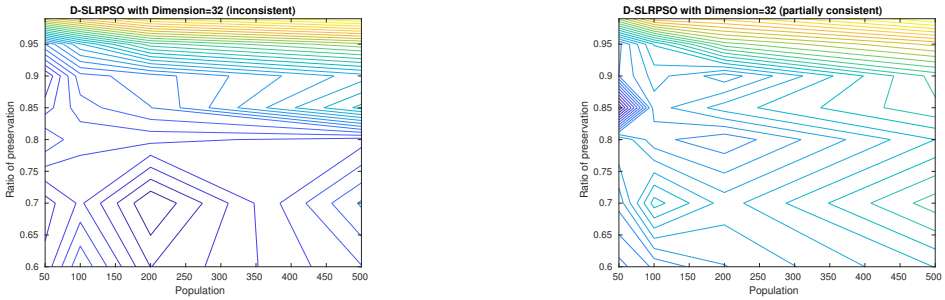


Figure 4.19: Ratio of Preservation vs. Size of Population (D-SLRPSO)

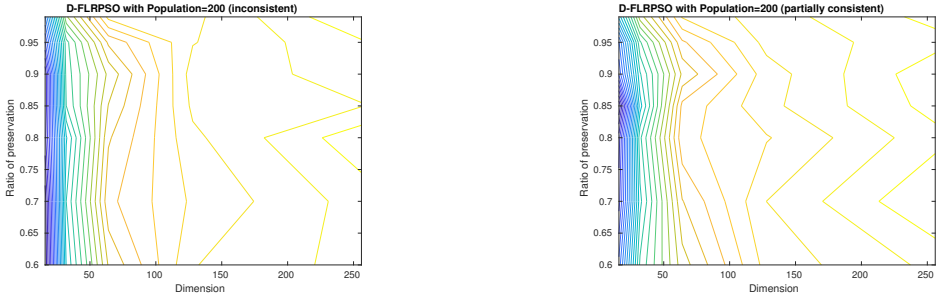


Figure 4.20: Ratio of Preservation vs. Dimension of Space (D-FLRPSO)

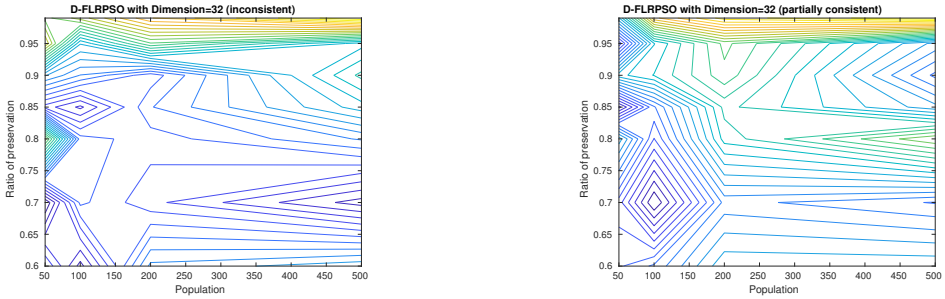


Figure 4.21: Ratio of Preservation vs. Size of Population (D-FLRPSO)

$= 16$, population $= 200$, iteration $= 400$ and the number of jobs $= 128$. For each pair we repeat the experiments three times and compute the average.

Table 4.24 - Table 4.26 show the rank of the pairs of (RP_{start}, RP_{end}) of D-SLRPSO for each ETC matrix. Table 4.27 is the summation of those ranks. The pairs $\{(0.8, 0.98)$, $(0.92, 0.98)$, $(0.96, 0.98)$, $(0.98, 0.98)$ $\}$ seem to be better than others. We will use the pair $(0.92, 0.98)$ for D-SLRPSO in the following experiments.

Table 4.24: Pair of (RP_{start}, RP_{end}) for D-SLRPSO (Consistent)

$RP_{start} \backslash RP_{end}$	0.8	0.9	0.92	0.94	0.96	0.98
0.8	15	6	2	1	3	8
0.9	-	10	9	16	20	19
0.92	-	-	11	12	13	5
0.94	-	-	-	4	17	21
0.96	-	-	-	-	18	14
0.98	-	-	-	-	-	7

Table 4.25: Pair of (RP_{start}, RP_{end}) for D-SLRPSO (Inconsistent)

$RP_{start} \backslash RP_{end}$	0.8	0.9	0.92	0.94	0.96	0.98
0.8	21	17	19	20	11	6
0.9	-	16	13	14	8	4
0.92	-	-	12	15	10	9
0.94	-	-	-	18	5	3
0.96	-	-	-	-	7	2
0.98	-	-	-	-	-	1

Table 4.26: Pair of (RP_{start}, RP_{end}) for D-SLRPSO (Partially Consistent)

$RP_{start} \backslash RP_{end}$	0.8	0.9	0.92	0.94	0.96	0.98
0.8	17	19	18	12	14	3
0.9	-	16	21	15	11	1
0.92	-	-	8	6	13	7
0.94	-	-	-	20	10	5
0.96	-	-	-	-	9	4
0.98	-	-	-	-	-	2

Table 4.27: Pair of (RP_{start}, RP_{end}) for D-SLRPSO (Sum)

$RP_{start} \backslash RP_{end}$	0.8	0.9	0.92	0.94	0.96	0.98
0.8	53	42	39	33	28	17
0.9	-	42	43	45	39	24
0.92	-	-	31	33	36	21
0.94	-	-	-	42	32	29
0.96	-	-	-	-	34	20
0.98	-	-	-	-	-	10

Table 4.28 - Table 4.30 show the rank of the pairs of (RP_{start}, RP_{end}) of D-FLRPSO for each ETC matrix. Table 4.31 is the summation of those ranks.
The pairs $\{(0.8, 0.98)$, $(0.92, 0.98)$, $(0.96, 0.98)\}$ seem to be better than others. We will use the pair $(0.92, 0.98)$ for D-FLRPSO in the following experiments.

Table 4.28: Pair of (RP_{start}, RP_{end}) for D-FLRPSO (Consistent)

$RP_{start} \backslash RP_{end}$	0.8	0.9	0.92	0.94	0.96	0.98
0.8	2	3	4	14	5	1
0.9	-	12	20	19	18	9
0.92	-	-	6	21	17	8
0.94	-	-	-	13	11	15
0.96	-	-	-	-	10	7
0.98	-	-	-	-	-	16

Table 4.29: Pair of (RP_{start}, RP_{end}) for D-FLRPSO (Inconsistent)

$RP_{start} \backslash RP_{end}$	0.8	0.9	0.92	0.94	0.96	0.98
0.8	21	18	19	16	14	11
0.9	-	20	17	13	12	5
0.92	-	-	15	9	7	4
0.94	-	-	-	10	8	6
0.96	-	-	-	-	3	2
0.98	-	-	-	-	-	1

Table 4.30: Pair of (RP_{start}, RP_{end}) for D-FLRPSO (Partially Consistent)

$RP_{start} \backslash RP_{end}$	0.8	0.9	0.92	0.94	0.96	0.98
0.8	20	19	15	11	18	3
0.9	-	12	17	16	13	10
0.92	-	-	14	21	5	6
0.94	-	-	-	7	8	1
0.96	-	-	-	-	9	2
0.98	-	-	-	-	-	4

Table 4.31: Pair of (RP_{start}, RP_{end}) for D-FLRPSO (Sum)

$RP_{start} \backslash RP_{end}$	0.8	0.9	0.92	0.94	0.96	0.98
0.8	43	40	38	41	37	15
0.9	-	44	54	48	43	24
0.92	-	-	35	51	29	18
0.94	-	-	-	30	27	22
0.96	-	-	-	-	22	11
0.98	-	-	-	-	-	21

Experimental Design

We compare the following four methods in the experiments of Independent Job Scheduling Problems.

1. **IDPSO** in Section 2.5.4:
   We follow the parameter specification in Izakian et al. [67]. $(c_1, c_2) = (2, 2)$. The maximum of the velocity is set to 40.

2. **D-SPSO** in Section 4.3.3:
   We set $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$ to meet the criteria in (4.3).

3. **D-SLRPSO** in Section 4.3.4:
   $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$ is set to meet the criteria in (4.3). We set $(RP_{start}, RP_{end}) = (0.92, 0.98)$ as specified in the preliminary experiments.

4. **D-FLRPSO** in Section 4.3.4:
   $(\omega, c_1, c_2, c_3) = (0.5, 1.3, 1.3, 1.3)$ is set to meet the criteria in (4.23). We set $(RP_{start}, RP_{end}) = (0.92, 0.98)$ as specified ed in the preliminary experiments.

The ETC matrices with 512 jobs and 16 machines are generated using the method in (Braun et al. [19]). There are 12 patterns of the ETC matrices which are labeled as x-yy-zz in Table 4.32.

- x denotes the type of consistency: C (consistent), I (inconsistent), P (partially consistent)
  C (consistent) means that if a machine m_i executes a job j faster then a machine m_k, the machine m_i execute all jobs faster than m_k. P (partially consistent) means that the ETC matrix includes a consistent sub-matrix of a predefined size.

- yy denotes the heterogeneity among jobs: Hi (high heterogeneity), Lo (low heterogeneity)

- zz denotes the heterogeneity among machines: Hi (high heterogeneity), Lo (low heterogeneity)

In the experiment, we set the size of population to 50 and the number of iteration to 2000. Experiments are repeated 50 times for each dataset.

Results

In Table 4.32, the best results are emphasized in boldface. We compare the performance of IDPSO and D-SPSO using t test. If the results between the two methods are significantly different (p-value < 0.05), (*) is shown beside the number in the column of the better result.

Table 4.32: Experiments of Independent Job Scheduling for Discrete PSO with Round Functions

Problem	**IDPSO**	**D-SPSO**	**D-SLRPSO**	**D-FLRPSO**
C-Hi-Hi	28608560.0 ± 1764133.3	**19247100.0*** ± 1662350.4	23211850.0 ± 1617517.3	23876730.0 ± 1234613.8
C-Hi-Lo	281377.2 ± 19187.3	**194886.5*** ± 16218.4	226902.9 ± 14508.3	238125.8 ± 12223.3
C-Lo-Hi	956887.1 ± 61952.6	**649563.3*** ± 56487.5	779415.7 ± 48856.0	798784.5 ± 40562.7
C-Lo-Lo	9408.4 ± 603.4	**6349.0*** ± 551.2	7744.1 ± 454.9	8022.4 ± 372.8
I-Hi-Hi	23855880.0 ± 996308.4	20194470.0* ± 891644.9	**19872710.0** ± 721753.3	20965150.0 ± 714236.2
I-Hi-Lo	238310.0 ± 10948.6	202185.0* ± 9638.2	**199797.7** ± 8889.4	211158.5 ± 8045.7
I-Lo-Hi	793423.2 ± 32887.9	668862.6* ± 28392.8	**663887.3** ± 29032.1	701331.7 ± 23086.8
I-Lo-Lo	7955.1 ± 335.7	6724.1* ± 285.0	**6640.0** ± 286.0	7037.8 ± 243.9
P-Hi-Hi	26008620.0 ± 1570246.5	**19444180.0*** ± 1097828.5	20588630.0 ± 1207435.4	21593420.0 ± 905985.2
P-Hi-Lo	261010.1 ± 15239.7	**194318.9*** ± 12476.0	203046.8 ± 10804.4	215973.4 ± 9734.7
P-Lo-Hi	864818.8 ± 48641.7	**644665.4*** ± 36199.5	683122.2 ± 37826.3	725966.6 ± 32832.6
P-Lo-Lo	8719.7 ± 464.9	**6545.3*** ± 363.9	6790.5 ± 407.9	7254.3 ± 292.8

Table 4.32 shows that all three proposed methods outperform IDPSO. We also compared the performance of IDPSO and D-SPSO using t test. For all problems D-SPSO are significantly better than IDPSO and the differences are quite large.

4.4.5 Experiments of Discrete PSO with Round Functions for Efficacy Improvement of Continuous PSO

As an another application of discrete PSO, we examine the efficacy improvement over continuous PSO using cache methods, which store the part of history in order to reduce the actual evaluation of fitness functions. Our goal is to verify that the discrete PSO with the round functions can save the computational cost by the cache methods while keeping the same level of accuracy as the continuous PSO.

As in the previous sections, firstly we optimize the perturbation parameters. Then we find the appropriate number of digits to the right of the decimal point to which the round function ψ rounds, so that the discrete PSO can achieve the same level of accuracy as the continuous PSO.

Preliminary Experiments

Specification of Perturbation Parameters: In this section we search for the optimal specification of perturbation parameter σ for D-SLRPSO and D-FLRPSO in Section 4.3.4. As stated in Section 4.3.5, we measure the difference of pbest before and after the perturbation as the ratio of perturbed distance to the whole space in (4.22):

$$RPD = \frac{\text{mean}(\|pbest2 - pbest\|)}{\sqrt{d'}c}.$$

where $\|\cdot\|$ is a Euclidean norm. We start with the experiments by visualizing how the optimal values of RPD change according to the different dimension of spaces and the different sizes of population. We conduct experiments with iteration $= 500$ using the benchmark function F4 and F6.

Figure 4.22 - Figure 4.25 are the contour maps of the optimal solutions for the input space of RPD ($\log_{1/10} RPD \in \{1, 2, 3, 4, 5\}$) vs. the size of dimension $\in \{10, 20, 30, 50\}$ and the RPD vs. the size of population $\in \{100, 200, 500, 1000\}$ for D-SLRPSO and D-FLRPSO. The left figure is the one for F4 and the right figure is the one for F6.

Next we determine the best combination of RPD. Based on Figure 4.22 - Figure 4.25, we choose the range around $\log_{1/10} RPD = 2$;

$$\log_{1/10} RPD \in \{1.5, 2, 2.5, 3, 3.5\} \tag{4.28}$$

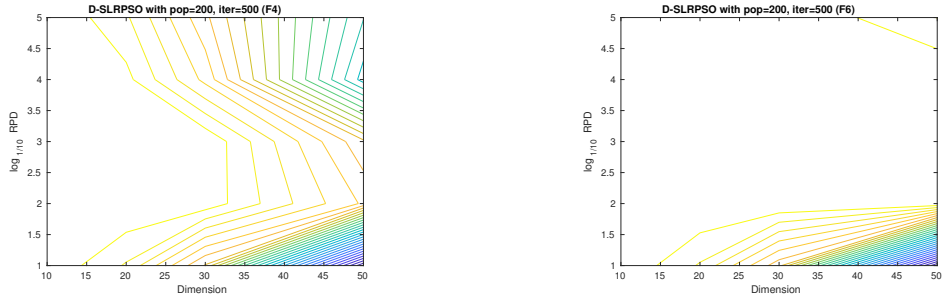


Figure 4.22: Ratio of Perturbed Distance vs. Dimension of Space (D-SLRPSO)

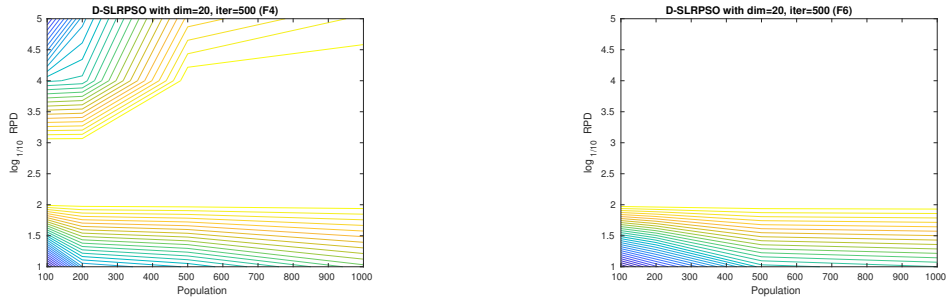


Figure 4.23: Ratio of Perturbed Distance vs. Size of Population (D-SLRPSO)

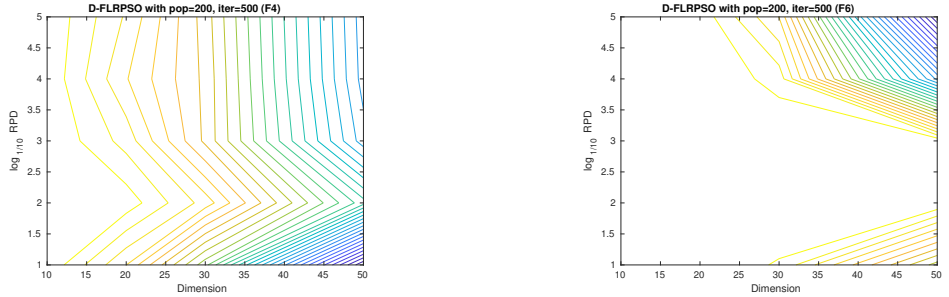


Figure 4.24: Ratio of Perturbed Distance vs. Dimension of Space (D-FLRPSO)

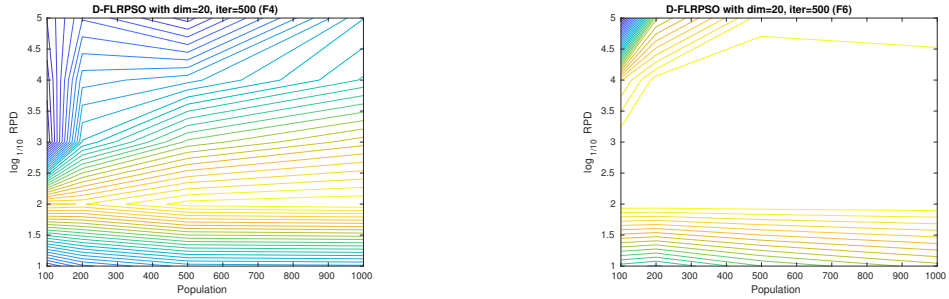


Figure 4.25: Ratio of Perturbed Distance vs. Size of Population (D-FLRPSO)

and decide the best pair of ($\log_{1/10} RPD_{start}$, $\log_{1/10} RPD_{end}$) from this range. We evaluate each pair using F4, F6 and F9. We set the dimension $= 20$, population $= 200$ and iteration $= 500$. For each pair we repeat the experiments three times and compute the average.

Table 4.33 - Table 4.35 show the rank of the pairs of ($\log_{1/10} RPD_{start}, \log_{1/10} RPD_{end}$) of D-SLRPSO for F4, F6 and F9. Table 4.36 is the summation of those ranks. From those tables, we choose the pair $(2, 3.5)$ for D-SLRPSO in the following experiments.

Table 4.33: Pair of ($\log_{1/10} RPD_{start}$, $\log_{1/10} RPD_{end}$) for D-SLRPSO (F4)

$\log_{1/10} RPD_{start} \backslash \log_{1/10} RPD_{end}$	1.5	2	2.5	3	3.5
1.5	15	14	12	8	7
2	-	13	10	6	3
2.5	-	-	9	4	1
3	-	-	-	5	2
3.5	-	-	-	-	11

Table 4.34: Pair of ($\log_{1/10} RPD_{start}$, $\log_{1/10} RPD_{end}$) for D-SLRPSO (F6)

$\log_{1/10} RPD_{start} \backslash \log_{1/10} RPD_{end}$	1.5	2	2.5	3	3.5
1.5	15	14	12	8	7
2	-	13	11	9	3
2.5	-	-	10	6	2
3	-	-	-	5	4
3.5	-	-	-	-	1

Table 4.35: Pair of ($\log_{1/10} RPD_{start}$, $\log_{1/10} RPD_{end}$) for D-SLRPSO (F9)

$\log_{1/10} RPD_{start} \backslash \log_{1/10} RPD_{end}$	1.5	2	2.5	3	3.5
1.5	14	11	3	1	2
2	-	12	4	6	5
2.5	-	-	10	8	15
3	-	-	-	7	13
3.5	-	-	-	-	9

Table 4.37 - Table 4.39 show the rank of the pairs of ($\log_{1/10} RPD_{start}, \log_{1/10} RPD_{end}$) of D-SLRPSO for F4, F6 and F9. Table 4.40 is the summation of those ranks. From those tables, we choose the pair $(1.5, 3.5)$ for D-FLRPSO in the following experiments.

Table 4.36: Pair of ($\log_{1/10} RPD_{start}$, $\log_{1/10} RPD_{end}$) for D-SLRPSO (Sum)

$\log_{1/10} RPD_{start} \backslash \log_{1/10} RPD_{end}$	1.5	2	2.5	3	3.5
1.5	44	39	27	17	16
2	-	38	25	21	11
2.5	-	-	29	18	18
3	-	-	-	17	19
3.5	-	-	-	-	21

Table 4.37: Pair of ($\log_{1/10} RPD_{start}$, $\log_{1/10} RPD_{end}$) for D-FLRPSO (F4)

$\log_{1/10} RPD_{start} \backslash \log_{1/10} RPD_{end}$	1.5	2	2.5	3	3.5
1.5	11	4	1	2	3
2	-	7	5	6	9
2.5	-	-	12	10	8
3	-	-	-	13	14
3.5	-	-	-	-	15

Table 4.38: Pair of ($\log_{1/10} RPD_{start}$, $\log_{1/10} RPD_{end}$) for D-FLRPSO (F6)

$\log_{1/10} RPD_{start} \backslash \log_{1/10} RPD_{end}$	1.5	2	2.5	3	3.5
1.5	15	14	8	7	2
2	-	12	9	6	3
2.5	-	-	10	4	1
3	-	-	-	5	11
3.5	-	-	-	-	13

Table 4.39: Pair of ($\log_{1/10} RPD_{start}$, $\log_{1/10} RPD_{end}$) for D-FLRPSO (F9)

$\log_{1/10} RPD_{start} \backslash \log_{1/10} RPD_{end}$	1.5	2	2.5	3	3.5
1.5	15	13	3	2	1
2	-	14	7	5	4
2.5	-	-	11	12	8
3	-	-	-	10	9
3.5	-	-	-	-	6

Determination of the calibration of round function: Next we will find the appropriate calibration of the round function so that the discrete PSO keep the same level of accuracy as the continuous PSO. Let denote the range of domain of the functions in

Table 4.40: Pair of ($\log_{1/10} RPD_{start}$, $\log_{1/10} RPD_{end}$) for D-FLRPSO (Sum)

$\log_{1/10} RPD_{start} \backslash \log_{1/10} RPD_{end}$	1.5	2	2.5	3	3.5
1.5	41	31	12	11	6
2	-	33	21	17	16
2.5	-	-	33	26	17
3	-	-	-	28	34
3.5	-	-	-	-	34

the last column of Table 4.1 [low, up]. and $\kappa = 10^{-\text{delN}}$. We set $delN$ to:

$$\text{delN} = |\lfloor \log_{10}(\text{up} - \text{low}) \rfloor - 4| \tag{4.29}$$

where $\lfloor z \rfloor$ is the largest integer $\leq z$. For instance, the range of domain of F1 is $[-100, 100]$ and so delN $= |\lfloor \log_{10}(200) \rfloor - 4| = 2$. Let calN be the number of digits to the right of the decimal point, to which the function ψ rounds. We select the appropriate calN from the range:

$$\text{calN} \in \{\text{delN} - 2, \text{delN} - 1.\text{delN}, \text{delN} + 1, \text{delN} + 2\} \tag{4.30}$$

In details, we conduct one-tailed t-test with significance level 0.15 with the alternative hypothesis that the mean fitness values of continuous PSO is smaller (better) than the mean fitness values of the discrete PSO. We identify the maximum digit maxN in (4.30) where the p-value is less than 0.15 and set calN to maxN $+ 1$.

For the preliminary experiments we set the number of iteration to 300 and the size of population to $100 \times (\text{dimension}/10)$ for the unimodal functions (F1, F2, F4, F5) and the number of iteration to 500 and the size of population to $200 \times (\text{dimension}/10)$ for the multi-modal benchmark functions (F6, F9, F10, F11, F12, F14). We repeat the experiments 30 times for each function. The results are shown in Table 4.41.

Experimental Design

Now we start the main experiments. Our goal is to verify that the discrete PSO with the round functions in (4.16) can save the computational cost by the cache methods while keeping the same level of accuracy as the continuous PSO. In the experiments we set the numbers of dimension to $(10, 20, 30)$ and set the sizes of cache to $1000, 1500, 2000$ instances for the dimension $10, 20, 30$, respectively. We compare the three pairs of continuous and discrete PSOs. The cache methods are applied to the discrete version of PSOs.

Table 4.41: Specification of calN

Func.	Dim.	D-SPSO	D-SLRPSO	D-FLRPSO
F1	10	3	2	2
	20	0	2	2
	30	0	2	2
F2	10	0	2	2
	20	0	2	5
	30	3	2	0
F4	10	0	2	1
	20	2	2	0
	30	0	0	0
F5	10	0	5	2
	20	0	0	0
	30	0	4	0
F6	10	0	1	2
	20	0	0	1
	30	0	0	2
F9	10	1	1	2
	20	1	2	2
	30	1	2	3
F10	10	1	2	2
	20	1	2	3
	30	1	2	2
F11	10	2	3	3
	20	3	3	3
	30	3	3	3
F12	10	1	2	5
	20	5	2	1
	30	1	2	0
F14	10	0	0	3
	20	0	0	0
	30	0	0	0

1. Standard PSO:

   - **SPSO** in Section 2.5.2:

We set $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$ to meet the criteria in (4.3).

- **D-SPSO** in Section 4.3.3:
  The parameter specification of (ω, c_1, c_2) is same as above.

2. Locally convergent rotationally invariant PSO with the "gbest" topology:

   - **SLcRiPSO** in Section 4.2.2:
     We set $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$ to meet the criteria in (4.3).
   - **D-SLRPSO** in Section 4.3.4:
     The parameter specification of (ω, c_1, c_2) is same as above. We set (RPD_{start}, RPD_{end})=(2,3.5) as determined in the preliminary experiments.

3. Locally convergent rotationally invariant PSO with the ring topology:

   - **FLcRiPSO** in Section 4.2.2:
     We set $(\omega, c_1, c_2, c_3) = (0.5, 1.3, 1.3, 1.3)$ is set to meet the criteria in (4.23).
   - **D-FLRPSO** in Section 4.3.4:
     The parameter specification of (ω, c_1, c_2, c_3) is same as above. We set (RPD_{start}, RPD_{end})=(1.5,3.5) as determined in the preliminary experiments.

Based on the parameters specified in the preliminary experiments, we conduct the experiments to compare the three pairs of continuous and discrete PSOs. We set the number of iteration to 500 and the size of population to $100\times$ (dimension $/10$) for the unimodal functions (F1, F2, F4, F5) and the number of iteration to 800 and the size of population to $200\times$ (dimension $/10$) for the multi-modal benchmark functions (F6, F9, F10, F11, F12, F14). We repeat the experiments 50 times for each function.

Results

In Table 4.42 the results of the continuous PSO and the discrete PSO are shown side by side. If the results of the discrete PSO are significantly different from those of the continuous PSO (p-values < 0.05 for the two-tailed t-test with unequal variance), (*) is shown beside the number in the column of the discrete PSO. The ratios of actual evaluation of fitness functions to total number of fitness evaluation in the discrete PSO are shown as the percentages in the parenthesis.

The objective of saving the computational cost without degrading the quality of solutions has been successfully achieved by D-SPSO. However, the savings of computational costs for D-SLRPSO and for D-FLRPSO were almost nothing. That is, the

Table 4.42: Experiments of Efficacy Improvement by Discrete PSO

Func.	Dim.	SPSO	D-SPSO	SLRPSO	D-SLRPSO	FLRPSO	D-FLRPSO
F1	10	378 ± 386	293 ± 312 (36%)	0.0675 ± 0.0225	0.0687 ± 0.0243 (100%)	0.111 ± 0.0306	0.109 ± 0.0277 (100%)
	20	2.23e+03 ± 1.36e+03	2.48e+03 ± 1.81e+03 (21%)	0.275 ± 0.0783	0.268 ± 0.0742 (100%)	0.826 ± 0.195	0.804 ± 0.157 (100%)
	30	6.25e+03 ± 3.22e+03	5.96e+03 ± 3.31e+03 (26%)	0.597 ± 0.143	0.579 ± 0.169 (100%)	2.86 ± 0.573	2.95 ± 0.537 (100%)
F2	10	131 ± 116	118 ± 110 (25%)	0.121 ± 0.0464	0.111 ± 0.0361 (100%)	0.458 ± 0.154	0.445 ± 0.152 (100%)
	20	928 ± 1.32e+03	$2.01e+03^*$ ± 2.03e+03 (63%)	1.2 ± 0.358	1.27 ± 0.404 (100%)	279 ± 83.1	270 ± 85.3 (100%)
	30	5.23e+03 ± 5.18e+03	5.54e+03 ± 6.04e+03 (100%)	6.98 ± 2.1	7.22 ± 2.26 (100%)	3.12e+03 ± 529	3.32e+03 ± 618 (100%)
F4	10	181 ± 266	143 ± 128 (47%)	0.15 ± 0.0631	0.152 ± 0.0666 (100%)	0.724 ± 0.241	0.494^* ± 0.182 (100%)
	20	1.49e+03 ± 2.31e+03	2.63e+03 ± 4.77e+03 (100%)	2.56 ± 1.13	2.36 ± 0.922 (100%)	1.31e+03 ± 432	1.19e+03 ± 486 (100%)
	30	9.3e+03 ± 7.47e+03	9.98e+03 ± 8.36e+03 (100%)	38.8 ± 18.1	119^* ± 75 (100%)	9.52e+03 ± 1.93e+03	9.43e+03 ± 1.81e+03 (100%)
F5	10	664 ± 1.85e+03	357 ± 1.2e+03 (8%)	8.52 ± 2.84	8.22 ± 2.2 (100%)	14.9 ± 3.6	14.7 ± 3.2 (100%)
	20	4.96e+03 ± 2.71e+03	4.67e+03 ± 2.25e+03 (44%)	958 ± 759	1.08e+03 ± 878 (100%)	1.23e+03 ± 379	1.35e+03 ± 365 (100%)
	30	9.91e+03 ± 2.57e+03	1.01e+04 ± 3.23e+03 (57%)	2.42e+03 ± 969	2.38e+03 ± 976 (100%)	2.84e+03 ± 282	2.79e+03 ± 365 (100%)
F6	10	3.75e+05 ± 7.46e+05	9.7e+05 ± 3.09e+06 (12%)	282 ± 1.26e+03	97.5 ± 190 (100%)	16.8 ± 2.91	16.4 ± 2.98 (100%)
	20	5.35e+07 ± 1.01e+08	3.09e+07 ± 3.09e+07 (17%)	219 ± 520	358 ± 541 (100%)	180 ± 206	131 ± 160 (100%)
	30	3.8e+08 ± 3.25e+08	3.69e+08 ± 3.93e+08 (23%)	2.76e+03 ± 4.26e+03	3.74e+03 ± 4.85e+03 (100%)	401 ± 126	416 ± 159 (100%)
F9	10	4.69 ± 3.99	7.87^* ± 6.32 (27%)	14.8 ± 6.53	20.5^* ± 8.57 (97%)	3.5 ± 1.16	3.37 ± 1 (100%)
	20	35.4 ± 16.3	42.2^* ± 14.1 (31%)	69.3 ± 22.1	67.6 ± 21.3 (100%)	17.4 ± 3.12	17.4 ± 3.34 (100%)
	30	78.6 ± 23.1	87.3 ± 27.9 (33%)	124 ± 30.6	130 ± 32.2 (100%)	39.7 ± 6.21	39.8 ± 6.46 (100%)
F10	10	20.6 ± 7.91	23.4 ± 8.92 (32%)	13.4 ± 5.34	11.2^* ± 5.28 (100%)	3.58 ± 0.96	3.68 ± 0.951 (100%)
	20	80.8 ± 23.9	80.7 ± 19.4 (30%)	52 ± 16.6	52.1 ± 17.8 (100%)	15.7 ± 2.78	16 ± 3.02 (100%)
	30	154 ± 42.3	148 ± 34.9 (37%)	105 ± 30.9	102 ± 27 (100%)	42.9 ± 5.96	42.8 ± 6.37 (100%)
F11	10	3.83 ± 1.43	4.62^* ± 1.52 (17%)	2.5 ± 1.48	2.17 ± 1.32 (100%)	0.929 ± 0.243	0.934 ± 0.164 (100%)
	20	12.8 ± 2.62	12.1 ± 3.01 (32%)	7.13 ± 2.29	7.47 ± 2.56 (100%)	6.03 ± 1.03	6.39 ± 0.938 (100%)
	30	21.4 ± 4.8	21.8 ± 4.27 (34%)	13.5 ± 3.97	13.5 ± 2.71 (100%)	13.6 ± 1.34	13.7 ± 1.3 (100%)
F12	10	1.34e+03 ± 1.79e+03	2.16e+03 ± 3.21e+03 (87%)	1.7e+03 ± 994	1.71e+03 ± 1.35e+03 (100%)	2.04e+04 ± 6.63e+03	1.89e+04 ± 6.25e+03 (100%)
	20	1.57e+04 ± 1.52e+04	1.31e+04 ± 1.27e+04 (97%)	2.8e+04 ± 1.46e+04	2.36e+04 ± 1.06e+04 (100%)	2.11e+05 ± 4.1e+04	1.96e+05 ± 4.46e+04 (100%)
	30	4.61e+04 ± 4.92e+04	6.64e+04 ± 5.65e+04 (86%)	9.78e+04 ± 4e+04	9.3e+04 ± 4.3e+04 (100%)	5.78e+05 ± 1.15e+05	$7.09e+05^*$ ± 1.2e+05 (100%)
F14	10	2.99 ± 0.441	3.17^* ± 0.455 (82%)	2.87 ± 0.576	3.09^* ± 0.445 (100%)	2.77 ± 0.243	2.74 ± 0.271 (100%)
	20	7.48 ± 0.546	7.39 ± 0.589 (97%)	7.61 ± 0.64	7.32^* ± 0.479 (100%)	7.39 ± 0.216	7.51^* ± 0.185 (100%)
	30	12.4 ± 0.565	12.4 ± 0.587 (98%)	12.2 ± 0.791	12.5^* ± 0.513 (100%)	12.5 ± 0.195	12.5 ± 0.194 (100%)

information in the cache was not used for D-SLRPSO and for D-FLRPSO. In real-world applications if the performance of SPSO is not different from that of SLRPSO and FLRPSO, the computational cost will be saved by applying D-SPSO.

4.4.6 Rotational Invariance Properties

In this section we conduct the experiments to check the rotational invariance property of D-SLRPSO and D-FLRPSO. The space is rotated by $(0, 5^\circ, \ldots, 180^\circ)$ and the Ellipse function in (Spears et al. [146]) are evaluated in the rotated space by the following methods.

- SPSO in Section 2.5.2
  We set $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$ to meet the criteria in (4.3)
- D-SPSO in Section 4.3.3
  $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$.
- D-SLRPSO in Section 4.3.4 $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$. We adaptively adjust the parameter σ in (4.18) so that $\log_{10} \sigma$ starts with -1 at the first iteration and

lineally decreases to -4.5 at the last iteration.

- FIPS in Section 2.5.2
  We set $(\omega, c_1, c_2, c_3) = (0.5, 1.3, 1.3, 1.3)$ is set to meet the criteria in (4.23).
- D-FLRPSO in Section 4.3.4 $(\omega, c_1, c_2, c_3) = (0.5, 1.3, 1.3, 1.3)$. We adaptively set σ in the same way as D-SLRPSO.

In the experiments, we set the number of dimension to 2, the number of iteration to 500 and the size of population to 20. The experiments are repeated 100 times.

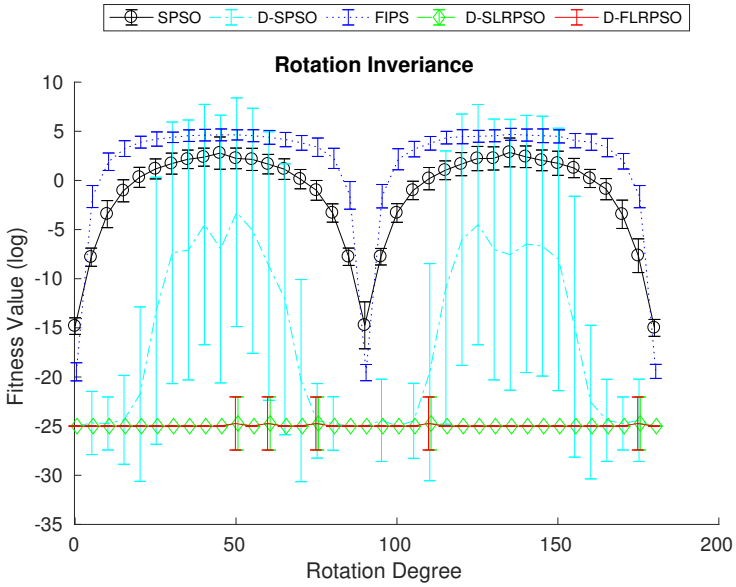


Figure 4.26: Experiment for Rotational Invariance Properties of D-SLRPSO and D-FLRPSO

4.4.7 Experiments for Gray versus Binary Encoding

In this section, we conduct experiments to test our claim that the Gray encoding is more appropriate than the binary encoding for the binary PSO.

Experimental Design

We compare the performance of the Gray encoding with the binary encoding using the following two binary PSO methods.

1. **B-SPSO** in Section 4.3.3:
   We set $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$ to meet the criteria in (4.3). We adaptively

adjust the parameter ξ in (4.17) so that ξ is 0.08 at the first iteration and lineally decreases to 0.01 at the last iteration.

2. **KBPSO** in Section 2.5.3:
   We follow the parameter specification in Khanesar et al. [75] and set $(\omega, c_1, c_2) = (0.5, 1, 1)$ and the maximum of the velocity $= 4$.

In the experiments, each real number in the input vector is represented by 20 bits of code. The dimension of real-valued solutions are set to be 3, 6 and 9 and so the dimension of the binary space is 60, 120 and 180, respectively.

We set both the number of iterations and the size of the population to $100\times$ (dimension of real solution $/3$) for the basic or unimodal functions (S1, S2, S3, S4, F1, F2, F4, F5) and the number of iterations to 500 and the size of the population to $300\times$ (dimension of real solution $/3$) for the multi-modal benchmark functions (F6, F9, F10, F11, F12, F14). The experiments are repeated 50 times for each function.

Results

The results of our experiments are presented in Table 4.43. For each method of B-SPSO and KBPSO we compare the Gray encoding with binary encoding. The better results are emphasized in boldface. We conduct the two-tailed t-test with unequal variance for each comparison. If the results of two encodings are significantly different (p-values < 0.05), (*) is shown beside the number in the column of the better encoding.

Tables 4.43 report the results of the experiments using B-SPSO and KBPSO. For B-SPSO, 19 out of 42 solutions using Gray encoding are significantly better ($p < 0.05$) than those using the binary encoding and 7 solutions using Gray encoding are significantly worse than those using the binary encoding. For KBPSO, 27 out of 42 solutions using Gray encoding are significantly better ($p < 0.05$) than those using the binary encoding and 6 solutions using Gray encoding are significantly worse than those using the binary encoding. In some cases (for instance, the F1, F2, F4, F6 functions), the difference is quite large.

4.5 Summary

In this chapter we proposed a unified approach of discretization of continuous PSO. In this approach discrete PSO models are just variants of continuous PSO and share the same velocity and position update formula. We presented general models in which

discrete PSO models are derived through the discretization functions. We introduced two types of discretization functions: a stochastic function for binary PSO, a round function for discrete PSO. We presented global and local convergence theorem for the discrete version of PSO. We also proposed a new method to find the optimal specifications of perturbation parameters for B(D)-SLRPSO and B(D)-FLRPSO.

For binary PSO, we showed in the experiments of benchmark functions and Knapsack Problems that the proposed "position as probability" approach outperforms the current "velocity as probability" approach. Therefore our conclusion is that there is no reason not to use the "position as probability" approach which is theoretically supported.

For discrete PSO, we showed that the discrete PSO with round functions can be used to solve the problems of the optimal sequences of integers with repetitions. We showed the possibility of efficacy improvement without degrading the quality of solutions for D-SPSO.

We will apply the discrete PSO to the hyper-parameter search in SVM using an adaptive calibration scheme to further improve the efficiency of continuous PSO in Chapter 5. We will also use the binary PSO for the feature subset selection in Chapter 7.

Contributions and achievements:

- We introduced a unified model in which binary and discrete PSO are variants of continuous PSO and the discrete versions of PSO are derived from the existing PSO. In this approach the discrete PSO benefits from various inventions and theoretical achievements in the continuous PSO.

- For the binary PSO we showed that the "position as probability" approach is not only theoretically supported but also outperforms the "velocity as probability" approach. As stated above we can expect a lot of improvements in this area because the binary PSO is now directly connected to the development of the continuous PSO.

- We also presented global and local convergence theorems for discrete versions of PSO in the proposed framework.

- We demonstrated the capabilities of the new discrete PSO in optimizing combinatorial problems by applying it to Independent Job Scheduling Problems.

- We proposed Gray code as an appropriate coding scheme of bit strings for the experiments of binary PSO using benchmark functions. We verified the claim in the experiments.

Table 4.43: Experiments for Gray versus Binary Encoding

Func.	Dim.	**B-SPSO** (Binary)	**B-SPSO** (Gray)	**KBPSO** (Binary)	**KBPSO**(Gray)
S1	3	**6.82e-09** ± 1.67e-24	**6.82e-09** ± 1.67e-24	**3.59e-05*** ± 5.54e-05	7.43e-05 ± 0.000124
	6	**1.36e-08** ± 1e-23	3.15e-08 ± 1e-07	**0.0192** ± 0.0153	0.0195 ± 0.0154
	9	**2.38e-07*** ± 2.14e-07	2.4e-06 ± 2.79e-06	0.519 ± 0.289	**0.349*** ± 0.229
S2	3	8.14 ± 11.3	**7.7** ± 9.63	5.42 ± 9.47	**4.43** ± 6.97
	6	57.3 ± 107	**8.89*** ± 9.83	51.8 ± 75.9	**17*** ± 20
	9	39.4 ± 81	**9.17*** ± 10.1	237 ± 267	**134*** ± 81.3
S3	3	**0.0153*** ± 0.0111	0.025 ± 0.0239	**0.0147** ± 0.0129	0.0164 ± 0.014
	6	**0.0348*** ± 0.0218	0.0488 ± 0.0334	**0.0608** ± 0.0344	0.0653 ± 0.0353
	9	**0.0622*** ± 0.0288	0.0746 ± 0.03	0.19 ± 0.0827	**0.122*** ± 0.047
S4	3	**0.319*** ± 0.781	1.65 ± 0.819	**0.632*** ± 0.728	1.57 ± 0.783
	6	**0.986*** ± 1.22	3.56 ± 1.24	8.68 ± 3.89	**4.91*** ± 1.69
	9	**3.42*** ± 2.49	5.63 ± 1.36	27.3 ± 7.87	**16.1*** ± 6.02
F1	3	4.89 ± 12.2	**4.21e-09*** ± 2.51e-24	1.5 ± 4.1	**0.000166*** ± 0.000342
	6	8.78 ± 17.4	**3.59e-08*** ± 7.62e-08	5.61 ± 6.39	**0.0484*** ± 0.0413
	9	22.8 ± 42.5	**4.02e-06*** ± 4.85e-06	41.2 ± 47.9	**1.06*** ± 0.735
F2	3	8.23 ± 18.8	**1.27e-07*** ± 2.84e-07	3.8 ± 12.3	**0.00152*** ± 0.00342
	6	20.8 ± 24.1	**0.000397*** ± 0.000551	16.9 ± 19.2	**0.726*** ± 0.67
	9	30.7 ± 31	**0.0378*** ± 0.0496	62.6 ± 43.2	**13.9*** ± 8.17
F4	3	7.38 ± 17.1	**6.39e-05*** ± 0.000437	2.19 ± 3.53	**0.00394*** ± 0.00772
	6	22.7 ± 22.5	**0.00772*** ± 0.0237	25.3 ± 28.2	**1.81*** ± 2.03
	9	88.6 ± 196	**0.438*** ± 1.12	90.7 ± 54.9	**22.5*** ± 25.1
F5	3	**3.93** ± 9.85	8.29 ± 13.2	**2.96*** ± 2.83	12.5 ± 12.3
	6	1.2 ± 0.656	**1.01** ± 0.872	**54.1*** ± 40.4	74.3 ± 44
	9	368 ± 206	**13.1*** ± 14.4	533 ± 210	**291*** ± 123
F6	3	45.8 ± 129	**6.3*** ± 11.1	20.1 ± 19.4	**2.6*** ± 5.98
	6	1.7e+03 ± 2.19e+03	**7.88*** ± 19.4	1.67e+03 ± 2.17e+03	**22.5*** ± 51.2
	9	3.07e+04 ± 1e+05	**16.7*** ± 29.2	3.48e+04 ± 1.02e+05	**110*** ± 109
F9	3	0.0912 ± 0.237	**0.0597** ± 0.239	0.0381 ± 0.0583	**3.23e-08*** ± 1.99e-07
	6	**0.306** ± 0.426	0.497 ± 0.611	0.991 ± 0.721	**0.567*** ± 0.746
	9	1.74 ± 1.1	**1.39** ± 1.1	4.65 ± 1.89	**2.53*** ± 1.12
F10	3	1.16 ± 1.08	**0.716*** ± 0.78	0.764 ± 0.652	**0.321*** ± 0.509
	6	**5.14** ± 2.76	5.57 ± 2.96	6.39 ± 2.9	**4.92*** ± 2.71
	9	12.9 ± 5	**11.1** ± 4.95	15.2 ± 5.41	**13.3** ± 6.27
F11	3	0.302 ± 0.219	**0.111*** ± 0.174	0.188 ± 0.176	**0.129** ± 0.145
	6	1.22 ± 0.589	**0.834*** ± 0.784	1.45 ± 0.66	**0.991*** ± 0.72
	9	3.39 ± 0.908	**2.45*** ± 0.994	3.93 ± 0.775	**3.28*** ± 0.96
F12	3	2.14 ± 7.16	**1.58** ± 6.26	**1.29** ± 5.35	1.35 ± 5.32
	6	27.7 ± 35.5	**17.4** ± 32.8	**39*** ± 36.4	76.6 ± 64
	9	**549** ± 802	552 ± 681	**715*** ± 716	1.7e+03 ± 823
F14	3	**0.0608** ± 0.0433	0.0847 ± 0.194	**0.056** ± 0.0354	0.0632 ± 0.139
	6	1.11 ± 0.414	**1.03** ± 0.443	1.07 ± 0.412	**1.05** ± 0.386
	9	2.45 ± 0.363	**2.34** ± 0.451	2.8 ± 0.257	**2.59*** ± 0.323

Chapter 5

Hyper-parameter Search using Discrete PSO

In Chapter 3, we proposed a new surface estimation method for the hyper-parameter search. The method constructed a surface (to estimate prediction accuracy) based on sample points on a regular grid. The surface was then used to identify the most promising regions in the hyper-parameter search space. The identified regions were then subject to a finer search to find the optimal values. We verified that this approach attained optimal or near optimal results on every dataset in the experiments. The method, however, has two limitations:

1. The range of the hyper-parameter search needs to be determined in the preliminary step.

2. Although the method requires significantly fewer sample points than similar methods (such as grid search), the number points required to construct the surface still grows exponentially with the increase in the number of hyper-parameters (i.e. the dimension of the search spaces).

The purpose for this chapter is to address these limitations.

5.1 Introduction

In this chapter, we examine an application of discrete PSO (Chapter 4) to hyper-parameter search in SVMs. We will show that the PSO approach provides a solution for the problems posed by the surface estimation methods in Chapter 3. Furthermore, in order to obtain optimal balance between accuracy and efficiency we propose a discrete PSO

with dynamic calibration of evaluation points. We present a simple algorithm to adaptively change the intervals between the evaluation points according to the density of particles. The algorithm enhances the efficiency of the standard PSO but not at the expense of accuracy.

We measure the improvement of efficiency based on the number of SVM runs instead of the actual running time, because actual running time of each SVM varies substantially according to the various factors such as the size of training data, datasets, types of kernels.

Gap or weakness in previous research: In the literature, PSO has been applied to the hyper-parameter search in SVMs but the existing methods are limited to two-dimensional hyper-parameter spaces. The reason is that higher dimensions require exponentially more evaluation points and this can be computationally infeasible because each evaluation requires a complete training of an SVM.

Objectives of Chapter 5: The goal is to use the properties of the new discrete PSO (proposed in the previous chapter) to conduct hyper-parameter search in higher dimensions and produce optimal solutions. The objective is to devise a system that can arrive at solutions found by SEB (a state-of-the-art method proposed in the previous chapter) and those methods using standard PSO, with significantly less computational effort.

5.2 Proposed Methods

We present an application of discrete PSO which adaptively changes the intervals of the evaluation points according to the density of particles. We also explain how to use PSO to automatically adjust search region.

5.2.1 Discrete PSO with Cellular Fitness Approximation

In Chapter 4 we considered discrete PSO using a general model. In the experiments we showed the possibility that the discrete PSO can improve the efficiency of continuous PSO without sacrificing accuracy. In this chapter we will go one step further and dynamically change the calibration of evaluation points. In order to change the intervals of evaluation points we can use two approaches:

1. to add a point at the middle of an interval between the adjacent evaluation points

2. to divide the space into cells and split a cell into subcells

The former is simpler, but in our experiments the latter approach generated better results. Therefore we propose the latter cell-based approach. The process is explained below.

We begin by partitioning the whole space into cells by dividing each axis into subintervals. Let the set $\{L_0^i, L_1^i, \ldots, L_{n_i}^i\}$ denote a partition of the x_i-axis into n_i subintervals, for $i \in \{1, \ldots, d'\}$, where $L_0^i < L_1^i < \cdots < L_{n_i}^i$, and $(L_0^i, L_{n_i}^i)$ includes the range of the i-th hyper-parameter. Then, for $k \in \{1, \ldots, n_i - 1\}$, the k-th subinterval I_k^i of the i-th axis, is defined as follows:

$$I_k^i = \{x : L_{k-1}^i \leq x < L_k^i\} \ ,$$

and the last, n_i-th, subinterval is defined as

$$I_{n_i}^i = \{x : L_{n_i-1}^i \leq x \leq L_{n_i}^i\} \ .$$

Then, a cell is defined as the Cartesian product of d' intervals, one along each of the d' input axes. Each particle in a given cell is evaluated at the center of that cell. Therefore, for a particle $(x_1, \ldots, x_{d'})$, where $x_i \in I_{k_i}^i$, for $k_i \in \{0, 1, \ldots, n_i\}$ and $i \in \{1, \ldots, d'\}$, the evaluation happens at the corresponding cell's center

$$\left(\frac{L_{k_1}^1 + L_{k_1+1}^1}{2}, \ldots, \frac{L_{k_{d'}}^{d'} + L_{k_{d'}+1}^{d'}}{2} \right) \ .$$

We keep a history of fitness computation at each cell. If the cell in which a particle arrives has been visited before, a cached fitness value is used instead of running SVM. Therefore we evaluate each cell at most once.

We change the size of cells dynamically according to the density of particles in order to conduct finer search in the important regions. For this purpose, we monitor the marginal distribution on each axis and count the number of particles in the slice on the interval I_k as depicted in Figure 5.1. If the accumulated number of particles exceeds some threshold value, we split the interval I_k into two subintervals by adding a point at the middle of the interval. The accumulated number of particles is recomputed in the divided intervals. Cached fitness values for the cells that have been divided become defunct and new fitness values will be computed as usual if necessary (Algorithm 5.1).Fitness Evaluation

We have used two types of cache process, one for recording marginal distribution on each axis and one for recording the history of position of particles. The marginal distribution on each axis can be recorded at a low cost. In the following experiments we set the minimum length of interval to 0.1. If the range of a parameter is $[-10, 10]$, for instance, we can store the information of marginal distribution for one parameter in an array of $\frac{20}{0.1} = 200$ rows. On the other hand, to record the history of positions of particles is an expensive process. In the experiments we will record history of all particles. However, it is impossible for larger experiments and we need to restrict the cache sizes in those cases.

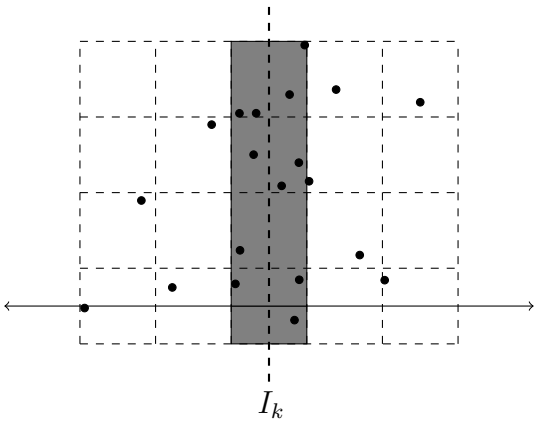


Figure 5.1: The slice on the interval I_k

The threshold values for splitting intervals should be determined based on the marginal distribution. For instance, we can use l times the mean of the number of particles in intervals on the axis. In order to avoid unnecessary precision and computational difficulties, each interval should keep a minimum width. We stop the splitting procedure if the length of intervals is less than some threshold value.

The PSO with cellular fitness approximation is implemented by the following algorithms.

5.2.2 Determination of Search Region

Before conducting a hyper-parameter search, we need to determine the search region. PSO can adjust the search region in an automatic fashion. For this purpose we use two boundaries, one for the boundary of the initial population and one for the boundary of particle movement. If the gbest goes outside the boundary of the initial population, we extend the boundaries. We keep a margin of fixed length ζ between the two

Algorithm 5.1: The process of PSO with cellular fitness approximation

Initialization:
1. initialize population position and velocity;
2. divide each axis by regular intervals;
3. add position of each particle to history of particle positions (HP);
4. set history of cells (HC) for fitness evaluation to [ ] (an empty array);
while termination conditions are not satisfied **do**
  1. call Algorithm 5.2 for fitness evaluation;
  2. call Algorithm 5.3 for re-partitioning of intervals;
  3. update pbest and gbest according to (2.32) and (2.33);
  4. update particle velocity according to (2.34);
  5. update particle position according to (2.31);
  if a particle position $\notin$ HP **then**
    Add the particle position to HP.
  end if
end while

Algorithm 5.2: Fitness evaluation

Require: HC, particle
determine the cell to which the particle belongs;
if the cell $\in$ HC **then**
  return the cached record of the fitness value;
else
  1. set the hyper-parameters of SVM to the center of the cell;
  2. learn a new SVM model based on the training data;
  3. run the model on the validation data;
  4. set the fitness to the accuracy of the model;
  5. add the cell and its fitness value to HC;
  return the fitness;
end if

Algorithm 5.3: Re-partitioning of intervals

Require: HP, particles

for each interval **do**

if the number of particles in the interval $>$ some threshold **and** the length of the interval $\geq$ some threshold **then**

1. split the interval in two halves;
2. re-compute the accumulated number of particles in the new intervals using HP

end if

end for

boundaries. The details of this procedure can be found in Algorithm 5.4.

5.3 Experiments

In this section, we conduct experiments with two objectives. The first objective is to verify the efficiency and accuracy of the standard PSO for the hyper-parameter search in SVMs by performing experiments in two-, three- and four-dimensional spaces. The second objective is to show that the enhancement of efficiency by the proposed method is achieved without sacrificing accuracy.

5.3.1 Experimental Design

Gaussian kernels, polynomial kernels with $b = 1$ and polynomial kernels in Section 2.3.3 are used for the two-, three- and four-dimensional experiments, respectively. It is meaningful since experiments for the hyper-parameter search in the literature are almost always restricted to two dimensional experiments. In each experiment, we search for the optimal specification of the hyper-parameters in the kernels and the regularization parameter C in (2.8). The initial ranges of the hyper-parameters are shown in the columns titled [LB, UB] and [GLB, GUB] in Table 5.1, whose entries are detailed settings of the hyper-parameters in the kernels and the regularization parameter. We set the ranges [LB, UB] and the grid size for SEB and GS as the same as in table 3.1 in Chapter 3.

We compare the prediction accuracy of the following three methods:

1. **SPSO** (the standard PSO):

Algorithm 5.4: Determination of search region

Require: Two boundaries [LB, UB] and [GLB, GUB] for each axis are given.

Set refining-boundaries = **true**.

while refining-boundaries **do**

1. initialize the population of PSO inside [LB, UB];
2. run PSO while particles are allowed to move inside [GLB, GUB];

if gbest $\leq$ LB **then**

1. set ζ = LB - GLB;
2. set LB $=$ GLB;
3. set GLB = GLB - ζ;

else if gbest $\geq$ UB **then**

1. set ζ = GUB - UB;
2. set UB $=$ GUB;
3. set GUB = GUB + ζ;

else

Set refining-boundaries = **false**.

end if

end while

return [LB, UB] and [GLB, GUB] for each axis.

Table 5.1: Initial settings of hyper-parameters

Kernel	Parameter	[LB, UB]	[GLB, GUB]	CFA-PSO Steps	SEB Grid Size	GS Grid Size	α
$\exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^2}\right)$	$\log\sigma^2$	$[-8,10]$	$[-10,14]$	2	2	$2/3$	20
Regularization parameter	$\log C$	$[-8,10]$	$[-10,12]$	2	2	$2/3$	
$(a\langle\mathbf{x}_i,\mathbf{x}_j\rangle+1)^c$	$\log a$	$[-8,10]$	$[-10,14]$	2	3	1	
	c	$[1,7]$	$[1,9]$	1	3	1	15
Regularization parameter	$\log C$	$[-8,10]$	$[-10,12]$	2	3	1	
$(a\langle\mathbf{x}_i,\mathbf{x}_j\rangle+b)^c$	$\log a$	$[-8,10]$	$[-10,14]$	2	3	1	
	b	$[0,9]$	$[0,12]$	1	3	1	10
	c	$[1,7]$	$[1,9]$	1	3	1	
Regularization parameter	$\log C$	$[-8,10]$	$[-10,12]$	2	3	1	

We set $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$ to meet the criteria laid out in [92]. We then examine four cases of population size and the number of generations as follows.

- **SPSO-10**: (population size = 10, number of generations = 10)
- **SPSO-20**: (population size = 20, number of generations = 20)
- **SPSO-30**: (population size = 30, number of generations = 30)
- **SPSO-40**: (population size = 40, number of generations = 40)

2. **CFA-PSO** (discrete PSO with cellular fitness approximation):
The values of (ω, c_1, c_2) are the same as above and four patterns of population size and the number of generations, **CFA-PSO-10**, **CFA-PSO-20**, **CFA-PSO-30** and **CFA-PSO-40** are prepared in the same way as SPSO. Initially the boundary of the initial population and the boundary of particle movement are set as in the "[LB, UB]" and the "[GLB, GUB]" column in Table 5.1. Within the boundary [GLB, GUB], the axis of each hyper-parameter is initially divided into regular intervals using the step size in the "CFA-PSO Steps" column. The initial search regions before running Algorithm 5.4 for Gaussian kernels and polynomial kernels and regularization parameter C are determined to follow the same setting in Chapter 3. After running Algorithm 5.4, the search regions are extended but the length of intervals between adjacent points is preserved. As the threshold value for splitting the intervals, we use three times the mean of the number of particles in the intervals. We stop the splitting procedure if the length of the intervals is less than 0.1.

3. **SEB** in Section 3.3 (Surface Estimation Method by Bézier curve):
   The initial search regions for SEB are determined by Algorithm 5.4. The step sizes are set as in the "SEB Grid Size" column in Table 5.1. For the second (finer) sampling, we set the size of the α-percent region to 20 for the two-dimensional experiments, to 15 for three-dimensional experiments and to 10 for the four-dimensional experiment as in the "α" column in Table 5.1.

4. **GS** in Section 3.3 (Grid Search with the finer grid sizes):
   The initial search regions for GS are determined by Algorithm 5.4. The step sizes are set as in the "GS Grid Size" column in Table 5.1, which are $1/3$ of the step sizes of SEB.

At first we run the SPSO and determine the search region using Algorithm 5.4. In order to avoid the computational difficulties due to the large values of hyper-parameters we extend the search regions only once in the experiment. Then we run SPSO, CFA-PSO, SEB and GS within the search region.

The efficiency of CFA-PSO compared to SEB is measured in terms of how many runs of SVM have been saved. More specifically, we define a metric, **Ratio**, to be the fraction of the number of runs of SVM in CFA-PSO to SEB. The lower the ratio, the higher the efficiency of the proposed method. These ratios for various problems and settings are later reported in Tables 5.2, 5.3 and 5.4.

We use the same benchmark datasets used in Chapter 3 and follow the same setting for the sizes of training, validation, and test datasets as specified in Table 1.2. Experiments are repeated 50 times for each dataset.

5.3.2 Results and Discussion

Tables 5.2, 5.3 and 5.4 report the results of binary classification carried out on the 17 benchmark datasets. In the last four columns of the tables, the first row in each cell shows the prediction accuracy of SPSO, the second row is the prediction accuracy of CFA-PSO and the third row shows the ratios of the number of SVM runs for CFA-PSO to that for SEB and to that for SPSO. We show the number of SVM runs for each method in the parenthesis. All the prediction accuracies are on the test datasets. The reported numbers are the average over 50 runs. We conduct the two-tailed paired t-test for each run of experiments to compare the prediction accuracy of SPSO and CFA-PSO to that of SEB. If the method is significantly better than SEB (p-values $<$ 0.05), (*) is shown beside the number in the column and if the method is significantly

worse than SEB (p-values < 0.05), ($^{-*}$) is shown beside the number in the column. We also conduct the two-tailed paired t-test to compare the prediction accuracy of SPSO to that of CFA-PSO. If SPSO is significantly better than CFA-PSO (p-values < 0.05), ($^{(*)}$) is shown beside the numbers in the row of SPSO, and if CFA-PSO is significantly better than SPSO, ($^{(*)}$) is shown beside the numbers in the row of CFA-PSO.

Table 5.2: Prediction accuracy and the ratio of actual runs of SVMs for two-dimensional experiments (using the Gaussian kernel)

Dataset	GS	SEB		PSO-10	PSO-20	PSO-30	PSO-40
Bank	99.50 ± 0.57	99.50 ± 0.58	**SPSO**	99.43 ± 0.63 (100)	99.53 ± 0.56 (400)	99.54 ± 0.57 (900)	$99.58^{(*)}$ ± 0.57 (1600)
	(794.1)	(239.1)	**CFA-PSO**	99.52 ± 0.59 (59.7)	99.51 ± 0.59 (122.8)	99.52 ± 0.59 (182.0)	99.51 ± 0.57 (240.8)
			Ratio	0.25(59.7/239.1), 0.60(59.7/100)	0.51(122.8/239.1), 0.31(122.8/400)	0.76(182.0/239.1), 0.20(182.0/900)	1.01(240.8/239.1), 0.15(240.8/1600)
Credit	80.88 ± 1.14	80.85 ± 1.12	**SPSO**	80.67 ± 1.09 (100)	80.88 ± 1.05 (400)	$80.93^{(*)}$ ± 1.01 (900)	80.89 ± 1.05 (1600)
	(821.3)	(352.8)	**CFA-PSO**	80.67^{-*} ± 1.09 (61.9)	80.86 ± 1.06 (127.4)	80.81 ± 0.99 (177.1)	80.87 ± 1.02 (225.6)
			Ratio	0.18(61.9/352.8), 0.62(61.9/100)	0.36(127.4/352.8), 0.32(127.4/400)	0.50(177.1/352.8), 0.20(177.1/900)	0.64(225.6/352.8), 0.14(225.6/1600)
Crowd	91.48 ± 1.11	91.48 ± 1.11	**SPSO**	91.39 ± 1.30 (100)	91.48 ± 1.09 (400)	91.52 ± 1.06 (900)	91.46 ± 1.08 (1600)
	(794.1)	(239.7)	**CFA-PSO**	91.38 ± 1.21 (60.4)	91.44 ± 1.05 (121.8)	91.52 ± 1.08 (178.3)	91.49 ± 1.08 (232.5)
			Ratio	0.25(60.4/239.7), 0.60(60.4/100)	0.51(121.8/239.7), 0.30(121.8/400)	0.74(178.3/239.7), 0.20(178.3/900)	0.97(232.5/239.7), 0.15(232.5/1600)
Drive	75.51 ± 2.16	75.51 ± 2.16	**SPSO**	75.59 ± 2.21 (100)	75.58 ± 2.17 (400)	75.64 ± 2.23 (900)	$75.64^{(*)}$ ± 2.28 (1600)
	(800.8)	(233.0)	**CFA-PSO**	75.50 ± 2.26 (59.4)	75.53 ± 2.19 (119.1)	75.52 ± 2.29 (174.8)	75.46 ± 2.24 (218.9)
			Ratio	0.26(59.4/233.0), 0.59(59.4/100)	0.51(119.1/233.0), 0.30(119.1/400)	0.75(174.8/233.0), 0.19(174.8/900)	0.94(218.9/233.0), 0.14(218.9/1600)
German	71.92 ± 2.65	71.92 ± 2.66	**SPSO**	71.81 ± 2.26 (100)	71.91 ± 2.57 (400)	71.59 ± 2.67 (900)	71.82 ± 2.66 (1600)
	(844.4)	(275.4)	**CFA-PSO**	$72.28^{(*)}$ ± 2.43 (62.8)	72.11 ± 2.57 (129.1)	71.88 ± 2.68 (167.3)	71.65 ± 2.87 (210.6)
			Ratio	0.23(62.8/275.4), 0.63(62.8/100)	0.47(129.1/275.4), 0.32(129.1/400)	0.61(167.3/275.4), 0.19(167.3/900)	0.76(210.6/275.4), 0.13(210.6/1600)
Letter	72.33 ± 2.19	72.33 ± 2.19	**SPSO**	72.18 ± 2.21 (100)	72.37 ± 2.17 (400)	72.37 ± 2.16 (900)	72.35 ± 2.13 (1600)
	(814.2)	(242.1)	**CFA-PSO**	72.16^{-*} ± 2.23 (59.4)	72.38 ± 2.18 (123.0)	72.34 ± 2.25 (174.2)	72.41 ± 2.27 (226.9)
			Ratio	0.25(59.4/242.1), 0.59(59.4/100)	0.51(123.0/242.1), 0.31(123.0/400)	0.72(174.2/242.1), 0.19(174.2/900)	0.94(226.9/242.1), 0.14(226.9/1600)
MAGIC	81.61 ± 1.72	81.61 ± 1.72	**SPSO**	81.54 ± 1.65 (100)	81.62 ± 1.70 (400)	81.66 ± 1.57 (900)	81.53 ± 1.64 (1600)
	(799.1)	(248.5)	**CFA-PSO**	81.48 ± 1.57 (59.2)	81.48 ± 1.70 (121.0)	81.54 ± 1.56 (180.4)	81.57 ± 1.67 (216.8)
			Ratio	0.24(59.2/248.5), 0.59(59.2/100)	0.49(121.0/248.5), 0.30(121.0/400)	0.73(180.4/248.5), 0.20(180.4/900)	0.87(216.8/248.5), 0.14(216.8/1600)
Occupancy	98.98 ± 0.20	98.98 ± 0.20	**SPSO**	98.97 ± 0.17 (100)	98.98 ± 0.20 (400)	98.98 ± 0.17 (900)	$99.00^{(*)}$ ± 0.15 (1600)
	(792.4)	(255.4)	**CFA-PSO**	98.95 ± 0.20 (60.0)	98.97 ± 0.21 (123.4)	98.97 ± 0.18 (173.6)	98.97 ± 0.20 (226.1)
			Ratio	0.23(60.0/255.4), 0.60(60.0/100)	0.48(123.4/255.4), 0.31(123.4/400)	0.68(173.6/255.4), 0.19(173.6/900)	0.89(226.1/255.4), 0.14(226.1/1600)
Opt Digit	95.51 ± 1.07	95.53 ± 1.02	**SPSO**	95.57 ± 1.03 (100)	95.58 ± 1.04 (400)	95.58 ± 1.03 (900)	95.58 ± 1.02 (1600)
	(789.0)	(213.8)	**CFA-PSO**	95.53 ± 1.02 (60.3)	95.55 ± 1.05 (124.7)	95.55 ± 1.03 (173.7)	95.56 ± 1.05 (230.3)
			Ratio	0.28(60.3/213.8), 0.60(60.3/100)	0.58(124.7/213.8), 0.31(124.7/400)	0.81(173.7/213.8), 0.19(173.7/900)	1.08(230.3/213.8), 0.14(230.3/1600)
Page Blocks	94.08 ± 0.91	94.08 ± 0.91	**SPSO**	94.10 ± 0.99 (100)	94.16 ± 0.97 (400)	94.18^{*} ± 0.98 (900)	94.18^{*} ± 0.93 (1600)
	(883.3)	(289.0)	**CFA-PSO**	94.10 ± 0.93 (64.7)	94.08 ± 0.96 (131.9)	94.17 ± 0.91 (201.7)	94.14 ± 0.90 (255.3)
			Ratio	0.22(64.7/289.0), 0.65(64.7/100)	0.46(131.9/289.0), 0.33(131.9/400)	0.70(201.7/289.0), 0.22(201.7/900)	0.88(255.3/289.0), 0.16(255.3/1600)
Pen Digit	95.22 ± 1.12	95.22 ± 1.12	**SPSO**	95.20 ± 1.06 (100)	95.29 ± 1.11 (400)	95.22 ± 1.09 (900)	95.26 ± 1.11 (1600)
	(795.8)	(220.9)	**CFA-PSO**	95.20 ± 1.14 (61.1)	95.24 ± 1.15 (125.5)	95.24 ± 1.10 (172.3)	95.25 ± 1.13 (231.6)
			Ratio	0.28(61.1/220.9), 0.61(61.1/100)	0.57(125.5/220.9), 0.31(125.5/400)	0.78(172.3/220.9), 0.19(172.3/900)	1.05(231.6/220.9), 0.14(231.6/1600)
Satellite	93.21 ± 0.70	93.18 ± 0.77	**SPSO**	93.19 ± 0.71 (100)	93.16 ± 0.66 (400)	93.20 ± 0.68 (900)	93.20 ± 0.64 (1600)
	(784.0)	(234.8)	**CFA-PSO**	93.09 ± 0.71 (58.1)	93.06^{-*} ± 0.78 (122.8)	93.18 ± 0.74 (172.2)	93.19 ± 0.70 (222.0)
			Ratio	0.25(58.1/234.8), 0.58(58.1/100)	0.52(122.8/234.8), 0.31(122.8/400)	0.73(172.2/234.8), 0.19(172.2/900)	0.95(222.0/234.8), 0.14(222.0/1600)
Segment	91.52 ± 1.34	91.52 ± 1.34	**SPSO**	91.45 ± 1.56 (100)	91.55 ± 1.43 (400)	91.58 ± 1.40 (900)	91.57 ± 1.40 (1600)
	(787.4)	(232.9)	**CFA-PSO**	91.43 ± 1.48 (59.5)	91.54 ± 1.40 (119.6)	91.49 ± 1.43 (167.3)	91.49 ± 1.41 (215.1)
			Ratio	0.26(59.5/232.9), 0.60(59.5/100)	0.51(119.6/232.9), 0.30(119.6/400)	0.72(167.3/232.9), 0.19(167.3/900)	0.92(215.1/232.9), 0.13(215.1/1600)
Splice	78.88 ± 1.53	78.88 ± 1.53	**SPSO**	78.90 ± 1.61 (100)	78.95 ± 1.64 (400)	78.94 ± 1.57 (900)	78.97 ± 1.54 (1600)
	(794.1)	(217.5)	**CFA-PSO**	78.82 ± 1.61 (60.1)	78.93 ± 1.51 (123.3)	78.92 ± 1.58 (179.6)	78.90 ± 1.65 (222.5)
			Ratio	0.28(60.1/217.5), 0.60(60.1/100)	0.57(123.3/217.5), 0.31(123.3/400)	0.83(179.6/217.5), 0.20(179.6/900)	1.02(222.5/217.5), 0.14(222.5/1600)
Statlog	98.45 ± 0.88	98.45 ± 0.87	**SPSO**	98.29^{-*} ± 0.89 (100)	98.44 ± 0.87 (400)	$98.51^{(*)}$ ± 0.83 (900)	$98.58^{*,(*)}$ ± 0.78 (1600)
	(807.5)	(254.9)	**CFA-PSO**	98.27^{-*} ± 0.83 (59.7)	98.41 ± 0.76 (126.2)	98.40 ± 0.82 (173.1)	98.42 ± 0.82 (227.5)
			Ratio	0.23(59.7/254.9), 0.60(59.7/100)	0.50(126.2/254.9), 0.32(126.2/400)	0.68(173.1/254.9), 0.19(173.1/900)	0.89(227.5/254.9), 0.14(227.5/1600)
Svmguide1	95.44 ± 0.68	95.44 ± 0.68	**SPSO**	95.38 ± 0.60 (100)	95.46 ± 0.60 (400)	95.42 ± 0.70 (900)	95.44 ± 0.54 (1600)
	(849.5)	(259.2)	**CFA-PSO**	95.44 ± 0.70 (61.9)	95.46 ± 0.71 (130.3)	95.44 ± 0.70 (193.6)	95.40 ± 0.68 (245.7)
			Ratio	0.24(61.9/259.2), 0.62(61.9/100)	0.50(130.3/259.2), 0.33(130.3/400)	0.75(193.6/259.2), 0.22(193.6/900)	0.95(245.7/259.2), 0.15(245.7/1600)
Wine Quality	79.52 ± 0.94	79.48 ± 0.96	**SPSO**	79.31^{-*} ± 0.86 (100)	79.43 ± 0.87 (400)	79.48 ± 0.93 (900)	79.44 ± 0.85 (1600)
	(795.8)	(288.6)	**CFA-PSO**	79.36 ± 0.87 (59.8)	79.42 ± 0.96 (119.7)	79.42 ± 0.92 (165.1)	79.49 ± 0.91 (211.7)
			Ratio	0.21(59.8/288.6), 0.60(59.8/100)	0.41(119.7/288.6), 0.30(119.7/400)	0.57(165.1/288.6), 0.18(165.1/900)	0.73(211.7/288.6), 0.13(211.7/1600)

We conducted two-, three- and four-dimensional experiments using two types of kernels which we have respectively reported in Tables 5.2, 5.3 and 5.4. First of all we notice that the overall performance of PSO is excellent. PSO-10 generated slightly worse but comparable results to SEB. In four dimensional experiments, the average

Table 5.3: Prediction accuracy and the ratio of actual runs of SVMs for three-dimensional experiments (using the Polynomial kernel with $b = 1$)

Dataset	GS	SEB		PSO-10	PSO-20	PSO-30	PSO-40
Bank	99.67 ± 0.37	99.55 ± 0.56	**SPSO**	99.53 ± 0.55 (100)	99.56 ± 0.50 (400)	99.63 ± 0.41 (900)	99.54 ± 0.53 (1600)
	(2596.5)	(690.4)	**CFA-PSO**	99.51 ± 0.42 (82.1)	99.54 ± 0.51 (218.8)	99.58 ± 0.45 (371.9)	99.59 ± 0.45 (520.4)
			Ratio	0.12(82.1/690.4), 0.82(82.1/100)	0.32(218.8/690.4), 0.55(218.8/400)	0.54(371.9/690.4), 0.41(371.9/900)	0.75(520.4/690.4), 0.33(520.4/1600)
Credit	81.10 ± 1.41	81.11 ± 1.44	**SPSO**	81.07 ± 1.42 (100)	81.10 ± 1.34 (400)	81.08 ± 1.44 (900)	81.04 ± 1.44 (1600)
	(2702.0)	(678.9)	**CFA-PSO**	80.79^{-*} ± 1.90 (78.1)	81.01 ± 1.41 (217.3)	81.08 ± 1.39 (357.9)	81.12 ± 1.41 (522.1)
			Ratio	0.12(78.1/678.9), 0.78(78.1/100)	0.32(217.3/678.9), 0.54(217.3/400)	0.53(357.9/678.9), 0.40(357.9/900)	0.77(522.1/678.9), 0.33(522.1/1600)
Crowd	89.06 ± 1.30	89.02 ± 1.28	**SPSO**	88.79^{-*} ± 1.44 (100)	88.79^{-*} ± 1.41 (400)	88.87 ± 1.33 (900)	88.92 ± 1.40 (1600)
	(2710.5)	(752.9)	**CFA-PSO**	88.73^{-*} ± 1.45 (81.4)	$89.03^{(*)}$ ± 1.27 (233.5)	88.93 ± 1.27 (393.1)	88.99 ± 1.29 (563.7)
			Ratio	0.11(81.4/752.9), 0.81(81.4/100)	0.31(233.5/752.9), 0.58(233.5/400)	0.52(393.1/752.9), 0.44(393.1/900)	0.75(563.7/752.9), 0.35(563.7/1600)
Drive	68.11 ± 2.56	67.90 ± 2.59	**SPSO**	67.81 ± 2.71 (100)	68.21^{*} ± 2.84 (400)	68.30^{*} ± 2.59 (900)	68.35^{*} ± 2.61 (1600)
	(3185.9)	(886.9)	**CFA-PSO**	67.77 ± 2.73 (82.5)	68.23^{*} ± 2.64 (236.4)	68.33^{*} ± 2.64 (388.1)	68.30^{*} ± 2.65 (579.6)
			Ratio	0.09(82.5/886.9), 0.83(82.5/100)	0.27(236.4/886.9), 0.59(236.4/400)	0.44(388.1/886.9), 0.43(388.1/900)	0.65(579.6/886.9), 0.36(579.6/1600)
German	71.88 ± 2.10	71.79 ± 2.06	**SPSO**	71.43 ± 2.11 (100)	71.77 ± 1.87 (400)	71.69 ± 2.06 (900)	71.89 ± 1.85 (1600)
	(2671.8)	(735.6)	**CFA-PSO**	70.92^{-*} ± 2.19 (78.9)	71.76 ± 2.03 (213.0)	71.90 ± 1.96 (395.8)	71.73 ± 2.02 (513.2)
			Ratio	0.11(78.9/735.6), 0.79(78.9/100)	0.29(213.0/735.6), 0.53(213.0/400)	0.54(395.8/735.6), 0.44(395.8/900)	0.70(513.2/735.6), 0.32(513.2/1600)
Letter	71.22 ± 2.15	71.11 ± 2.09	**SPSO**	71.23 ± 2.05 (100)	71.31 ± 1.99 (400)	71.19 ± 2.01 (900)	71.43^{*} ± 2.01 (1600)
	(2699.3)	(743.1)	**CFA-PSO**	70.96 ± 2.06 (79.6)	71.13 ± 2.08 (231.4)	71.16 ± 2.04 (382.3)	71.25 ± 2.04 (513.5)
			Ratio	0.11(79.6/743.1), 0.80(79.6/100)	0.31(231.4/743.1), 0.58(231.4/400)	0.51(382.3/743.1), 0.42(382.3/900)	0.69(513.5/743.1), 0.32(513.5/1600)
MAGIC	81.62 ± 1.46	81.55 ± 1.49	**SPSO**	81.46 ± 1.56 (100)	81.52 ± 1.49 (400)	81.56 ± 1.50 (900)	81.68 ± 1.45 (1600)
	(2784.6)	(744.7)	**CFA-PSO**	81.41 ± 1.52 (81.4)	81.47 ± 1.44 (241.9)	81.65 ± 1.47 (390.0)	81.72^{*} ± 1.41 (559.0)
			Ratio	0.11(81.4/744.7), 0.81(81.4/100)	0.32(241.9/744.7), 0.60(241.9/400)	0.52(390.0/744.7), 0.43(390.0/900)	0.75(559.0/744.7), 0.35(559.0/1600)
Occupancy	98.57 ± 0.30	98.56 ± 0.29	**SPSO**	98.58 ± 0.31 (100)	98.55 ± 0.34 (400)	98.58 ± 0.31 (900)	98.57 ± 0.33 (1600)
	(2738.6)	(785.5)	**CFA-PSO**	98.54 ± 0.37 (77.0)	98.58 ± 0.28 (206.0)	98.56 ± 0.31 (337.6)	98.57 ± 0.32 (447.8)
			Ratio	0.10(77.0/785.5), 0.77(77.0/100)	0.26(206.0/785.5), 0.52(206.0/400)	0.43(337.6/785.5), 0.38(337.6/900)	0.57(447.8/785.5), 0.28(447.8/1600)
Opt Digit	95.29 ± 1.00	95.28 ± 1.01	**SPSO**	95.29 ± 1.02 (100)	95.29 ± 0.99 (400)	95.28 ± 1.01 (900)	95.31 ± 0.99 (1600)
	(2918.0)	(870.4)	**CFA-PSO**	95.28 ± 0.99 (82.4)	95.29 ± 1.01 (230.2)	95.29 ± 1.00 (372.5)	95.27 ± 1.02 (494.2)
			Ratio	0.09(82.4/870.4), 0.82(82.4/100)	0.26(230.2/870.4), 0.58(230.2/400)	0.43(372.5/870.4), 0.41(372.5/900)	0.57(494.2/870.4), 0.31(494.2/1600)
Page Blocks	94.43 ± 0.64	94.39 ± 0.61	**SPSO**	94.32 ± 0.77 (100)	94.36 ± 0.62 (400)	94.39 ± 0.66 (900)	94.38 ± 0.68 (1600)
	(2697.0)	(790.1)	**CFA-PSO**	94.26 ± 0.69 (78.1)	94.37 ± 0.65 (215.2)	94.41 ± 0.69 (345.8)	94.40 ± 0.64 (462.4)
			Ratio	0.10(78.1/790.1), 0.78(78.1/100)	0.27(215.2/790.1), 0.54(215.2/400)	0.44(345.8/790.1), 0.38(345.8/900)	0.59(462.4/790.1), 0.29(462.4/1600)
Pen Digit	95.52 ± 0.98	95.44 ± 1.02	**SPSO**	95.39 ± 1.01 (100)	95.42 ± 1.04 (400)	95.43 ± 0.97 (900)	95.45 ± 1.02 (1600)
	(3000.1)	(851.2)	**CFA-PSO**	95.38 ± 1.08 (83.3)	95.41 ± 1.00 (238.1)	95.47 ± 0.98 (397.0)	95.44 ± 1.06 (547.3)
			Ratio	0.10(83.3/851.2), 0.83(83.3/100)	0.28(238.1/851.2), 0.60(238.1/400)	0.47(397.0/851.2), 0.44(397.0/900)	0.64(547.3/851.2), 0.34(547.3/1600)
Satellite	92.81 ± 0.61	92.80 ± 0.59	**SPSO**	92.63^{-*} ± 0.78 (100)	92.71 ± 0.73 (400)	92.72 ± 0.72 (900)	92.82 ± 0.58 (1600)
	(2613.6)	(684.4)	**CFA-PSO**	92.59^{-*} ± 0.86 (80.0)	92.74 ± 0.67 (224.0)	92.74 ± 0.66 (392.2)	92.82 ± 0.59 (519.7)
			Ratio	0.12(80.0/684.4), 0.80(80.0/100)	0.33(224.0/684.4), 0.56(224.0/400)	0.57(392.2/684.4), 0.44(392.2/900)	0.76(519.7/684.4), 0.32(519.7/1600)
Segment	91.35 ± 1.24	91.33 ± 1.14	**SPSO**	90.85^{-*} ± 1.49 (100)	91.13^{-*} ± 1.33 (400)	91.42 ± 1.31 (900)	91.38 ± 1.21 (1600)
	(3141.1)	(898.6)	**CFA-PSO**	91.04^{-*} ± 1.39 (82.5)	91.14 ± 1.35 (243.5)	91.26 ± 1.23 (430.6)	91.31 ± 1.22 (603.1)
			Ratio	0.09(82.5/898.6), 0.82(82.5/100)	0.27(243.5/898.6), 0.61(243.5/400)	0.48(430.6/898.6), 0.48(430.6/900)	0.67(603.1/898.6), 0.38(603.1/1600)
Splice	79.16 ± 1.55	79.09 ± 1.49	**SPSO**	78.73^{-*} ± 1.61 (100)	78.90 ± 1.53 (400)	78.97 ± 1.53 (900)	79.09 ± 1.44 (1600)
	(2659.2)	(577.0)	**CFA-PSO**	78.86 ± 1.47 (80.3)	79.03 ± 1.51 (229.9)	79.13 ± 1.53 (380.0)	79.12 ± 1.53 (525.1)
			Ratio	0.14(80.3/577.0), 0.80(80.3/100)	0.40(229.9/577.0), 0.57(229.9/400)	0.66(380.0/577.0), 0.42(380.0/900)	0.91(525.1/577.0), 0.33(525.1/1600)
Statlog	98.03 ± 1.03	97.98 ± 1.02	**SPSO**	97.87^{-*} ± 1.01 (100)	97.98 ± 1.09 (400)	98.09^{*} ± 1.02 (900)	98.06 ± 1.08 (1600)
	(2936.3)	(804.6)	**CFA-PSO**	97.74^{-*} ± 1.12 (81.8)	98.01 ± 0.97 (234.2)	98.03 ± 1.01 (394.0)	98.07 ± 0.99 (557.7)
			Ratio	0.10(81.8/804.6), 0.82(81.8/100)	0.29(234.2/804.6), 0.59(234.2/400)	0.49(394.0/804.6), 0.44(394.0/900)	0.69(557.7/804.6), 0.35(557.7/1600)
Svmguide1	95.64 ± 0.58	95.60 ± 0.59	**SPSO**	95.41^{-*} ± 0.68 (100)	95.53 ± 0.56 (400)	95.59 ± 0.54 (900)	95.58 ± 0.62 (1600)
	(2873.8)	(807.1)	**CFA-PSO**	95.42^{-*} ± 0.63 (80.1)	95.53 ± 0.60 (225.4)	95.58 ± 0.57 (389.5)	95.59 ± 0.59 (517.3)
			Ratio	0.10(80.1/807.1), 0.80(80.1/100)	0.28(225.4/807.1), 0.56(225.4/400)	0.48(389.5/807.1), 0.43(389.5/900)	0.64(517.3/807.1), 0.32(517.3/1600)
Wine Quality	79.28 ± 0.87	79.20 ± 0.90	**SPSO**	79.15 ± 0.88 (100)	79.28 ± 0.84 (400)	79.20 ± 0.81 (900)	79.22 ± 0.86 (1600)
	(2854.8)	(747.1)	**CFA-PSO**	79.11 ± 0.89 (82.0)	79.19 ± 0.88 (228.7)	79.28 ± 0.77 (408.2)	79.22 ± 0.84 (557.2)
			Ratio	0.11(82.0/747.1), 0.82(82.0/100)	0.31(228.7/747.1), 0.57(228.7/400)	0.55(408.2/747.1), 0.45(408.2/900)	0.75(557.2/747.1), 0.35(557.2/1600)

runs of SVMs for SEB is about 5500 as explained below , but the total runs of SVMs for PSO-10 is only 100. Computational complexity of PSO increases linearly with respect to the increase of dimensionality of spaces. From those results we hope that PSO can be applicable in higher dimensional spaces.

All tables indicate that the prediction accuracy of CFA-PSO-30 and CFA-PSO-40 are equivalent to that of SEB. In Table 5.2, CFA-PSO-30 nor CFA-PSO-40 is not significantly different from SEB for all datasets. In Table 5.3, CFA-PSO-30 is significantly better than SEB for one dataset, and CFA-PSO-40 is significantly better than SEB for two datasets. In Table 5.4, CFA-PSO-30 is significantly worse then SEB for three datasets, and CFA-

Table 5.4: Prediction accuracy and the ratio of actual runs of SVMs for four-dimensional experiments (using the polynomial kernel)

Dataset	GS	SEB		PSO-10	PSO-20	PSO-30	PSO-40
Bank	99.51 ± 0.47 (25900.4)	99.48 ± 0.48 (4984.5)	**SPSO**	99.49 ± 0.45 (100)	99.55 ± 0.43 (400)	99.49 ± 0.48 (900)	99.55 ± 0.50 (1600)
			CFA-PSO	99.43 ± 0.61 (88.7)	99.45 ± 0.59 (286.0)	99.52 ± 0.46 (544.8)	99.51 ± 0.48 (844.9)
			Ratio	0.02(88.7/4984.5), 0.89(88.7/100)	0.06(286.0/4984.5), 0.71(286.0/400)	0.11(544.8/4984.5), 0.61(544.8/900)	0.17(844.9/4984.5), 0.53(844.9/1600)
Credit	81.33 ± 1.27 (28870.7)	81.33 ± 1.22 (5007.4)	**SPSO**	81.27 ± 1.25 (100)	81.41 ± 1.30 (400)	81.36 ± 1.30 (900)	81.33 ± 1.24 (1600)
			CFA-PSO	81.24 ± 1.30 (88.8)	81.39 ± 1.24 (297.8)	81.37 ± 1.31 (567.6)	$81.40^{(*)} \pm 1.24$ (851.6)
			Ratio	0.02(88.8/5007.4), 0.89(88.8/100)	0.06(297.8/5007.4), 0.74(297.8/400)	0.11(567.6/5007.4), 0.63(567.6/900)	0.17(851.6/5007.4), 0.53(851.6/1600)
Crowd	89.07 ± 1.29 (27326.6)	88.84 ± 1.37 (5267.3)	**SPSO**	$88.22^{-*} \pm 1.56$ (100)	$88.43^{-*} \pm 1.54$ (400)	88.93 ± 1.34 (900)	88.86 ± 1.29 (1600)
			CFA-PSO	$88.25^{-*} \pm 1.47$ (89.2)	88.71 ± 1.37 (304.5)	88.77 ± 1.23 (556.6)	88.82 ± 1.27 (858.6)
			Ratio	0.02(89.2/5267.3), 0.89(89.2/100)	0.06(304.5/5267.3), 0.76(304.5/400)	0.11(556.6/5267.3), 0.62(556.6/900)	0.16(858.6/5267.3), 0.54(858.6/1600)
Drive	67.83 ± 3.00 (32121.5)	67.41 ± 2.75 (6304.5)	**SPSO**	67.35 ± 2.99 (100)	67.45 ± 2.93 (400)	$67.76^{*} \pm 2.56$ (900)	$68.02^{*} \pm 2.82$ (1600)
			CFA-PSO	67.20 ± 2.93 (90.0)	67.43 ± 2.91 (292.2)	67.47 ± 2.81 (547.1)	$67.76^{*} \pm 2.71$ (827.2)
			Ratio	0.01(90.0/6304.5), 0.90(90.0/100)	0.05(292.2/6304.5), 0.73(292.2/400)	0.09(547.1/6304.5), 0.61(547.1/900)	0.13(827.2/6304.5), 0.52(827.2/1600)
German	72.06 ± 1.69 (28451.1)	72.14 ± 1.72 (5637.6)	**SPSO**	71.80 ± 2.23 (100)	72.00 ± 1.81 (400)	72.26 ± 1.93 (900)	72.27 ± 1.88 (1600)
			CFA-PSO	71.84 ± 2.31 (88.0)	72.08 ± 2.06 (273.7)	72.27 ± 1.98 (543.9)	72.17 ± 1.75 (816.2)
			Ratio	0.02(88.0/5637.6), 0.88(88.0/100)	0.05(273.7/5637.6), 0.68(273.7/400)	0.10(543.9/5637.6), 0.60(543.9/900)	0.14(816.2/5637.6), 0.51(816.2/1600)
Letter	71.51 ± 2.33 (29361.9)	71.30 ± 2.03 (5809.7)	**SPSO**	$70.82^{-*} \pm 2.11$ (100)	71.19 ± 2.28 (400)	71.26 ± 2.26 (900)	71.13 ± 2.03 (1600)
			CFA-PSO	$70.96^{-*} \pm 2.17$ (89.7)	71.19 ± 2.04 (299.4)	71.06 ± 1.88 (550.7)	71.26 ± 2.27 (850.3)
			Ratio	0.02(89.7/5809.7), 0.90(89.7/100)	0.05(299.4/5809.7), 0.75(299.4/400)	0.09(550.7/5809.7), 0.61(550.7/900)	0.15(850.3/5809.7), 0.53(850.3/1600)
MAGIC	81.87 ± 1.41 (28410.7)	81.95 ± 1.26 (5449.3)	**SPSO**	$81.20^{-*} \pm 1.43$ (100)	$81.58^{-*} \pm 1.50$ (400)	$81.61^{-*} \pm 1.42$ (900)	81.82 ± 1.47 (1600)
			CFA-PSO	$81.46^{-*,(*)} \pm 1.40$ (90.4)	$81.51^{-*} \pm 1.44$ (301.4)	$81.54^{-*} \pm 1.37$ (549.4)	$81.64^{-*} \pm 1.43$ (855.0)
			Ratio	0.02(90.4/5449.3), 0.90(90.4/100)	0.06(301.4/5449.3), 0.75(301.4/400)	0.10(549.4/5449.3), 0.61(549.4/900)	0.16(855.0/5449.3), 0.53(855.0/1600)
Occupancy	98.60 ± 0.30 (28125.2)	98.62 ± 0.30 (5884.7)	**SPSO**	98.55 ± 0.40 (100)	98.59 ± 0.29 (400)	98.63 ± 0.29 (900)	98.63 ± 0.28 (1600)
			CFA-PSO	98.58 ± 0.31 (89.2)	98.55 ± 0.39 (284.7)	98.59 ± 0.33 (511.8)	98.60 ± 0.31 (811.9)
			Ratio	0.02(89.2/5884.7), 0.89(89.2/100)	0.05(284.7/5884.7), 0.71(284.7/400)	0.09(511.8/5884.7), 0.57(511.8/900)	0.14(811.9/5884.7), 0.51(811.9/1600)
Opt Digit	95.27 ± 1.05 (29060.6)	95.26 ± 1.07 (6058.0)	**SPSO**	95.22 ± 1.07 (100)	95.20 ± 1.08 (400)	95.23 ± 1.04 (900)	95.25 ± 1.04 (1600)
			CFA-PSO	95.27 ± 1.04 (91.0)	95.20 ± 1.06 (292.3)	95.21 ± 1.08 (559.2)	95.24 ± 1.05 (851.9)
			Ratio	0.02(91.0/6058.0), 0.91(91.0/100)	0.05(292.3/6058.0), 0.73(292.3/400)	0.09(559.2/6058.0), 0.62(559.2/900)	0.14(851.9/6058.0), 0.53(851.9/1600)
Page Blocks	94.65 ± 0.84 (29808.1)	94.58 ± 0.82 (6205.2)	**SPSO**	$94.35^{-*} \pm 0.93$ (100)	$94.56^{(*)} \pm 0.80$ (400)	94.56 ± 0.79 (900)	94.58 ± 0.81 (1600)
			CFA-PSO	$94.26^{-*} \pm 0.90$ (88.3)	$94.40^{-*} \pm 0.88$ (293.5)	94.51 ± 0.82 (538.0)	94.59 ± 0.81 (838.3)
			Ratio	0.01(88.3/6205.2), 0.88(88.3/100)	0.05(293.5/6205.2), 0.73(293.5/400)	0.09(538.0/6205.2), 0.60(538.0/900)	0.14(838.3/6205.2), 0.52(838.3/1600)
Pen Digit	95.45 ± 1.26 (30092.2)	95.43 ± 1.27 (5966.3)	**SPSO**	95.35 ± 1.28 (100)	95.41 ± 1.31 (400)	95.46 ± 1.25 (900)	95.46 ± 1.26 (1600)
			CFA-PSO	95.36 ± 1.30 (90.7)	95.40 ± 1.28 (293.2)	95.44 ± 1.29 (566.2)	95.46 ± 1.25 (865.5)
			Ratio	0.02(90.7/5966.3), 0.91(90.7/100)	0.05(293.2/5966.3), 0.73(293.2/400)	0.09(566.2/5966.3), 0.63(566.2/900)	0.15(865.5/5966.3), 0.54(865.5/1600)
Satellite	92.65 ± 0.56 (27513.5)	92.54 ± 0.60 (5309.6)	**SPSO**	$92.30^{-*} \pm 0.77$ (100)	$92.37^{-*} \pm 0.77$ (400)	92.58 ± 0.64 (900)	92.52 ± 0.66 (1600)
			CFA-PSO	$92.25^{-*} \pm 0.83$ (88.4)	$92.57^{(*)} \pm 0.70$ (294.2)	92.51 ± 0.74 (554.8)	92.59 ± 0.75 (848.7)
			Ratio	0.02(88.4/5309.6), 0.88(88.4/100)	0.06(294.2/5309.6), 0.74(294.2/400)	0.10(554.8/5309.6), 0.62(554.8/900)	0.16(848.7/5309.6), 0.53(848.7/1600)
Segment	91.42 ± 1.49 (30218.8)	91.36 ± 1.46 (6185.0)	**SPSO**	$90.96^{-*} \pm 1.78$ (100)	$91.09^{-*} \pm 1.54$ (400)	91.18 ± 1.62 (900)	$91.18^{-*} \pm 1.72$ (1600)
			CFA-PSO	$90.91^{-*} \pm 1.84$ (90.8)	$91.16^{-*} \pm 1.66$ (292.5)	$91.17^{-*} \pm 1.67$ (563.6)	91.18 ± 1.54 (851.6)
			Ratio	0.01(90.8/6185.0), 0.91(90.8/100)	0.05(292.5/6185.0), 0.73(292.5/400)	0.09(563.6/6185.0), 0.63(563.6/900)	0.14(851.6/6185.0), 0.53(851.6/1600)
Splice	78.89 ± 1.56 (28523.6)	78.65 ± 1.58 (4623.9)	**SPSO**	78.69 ± 1.66 (100)	78.70 ± 1.59 (400)	78.68 ± 1.62 (900)	78.74 ± 1.67 (1600)
			CFA-PSO	78.52 ± 1.64 (90.4)	78.65 ± 1.64 (298.3)	78.80 ± 1.56 (566.5)	78.75 ± 1.56 (822.6)
			Ratio	0.02(90.4/4623.9), 0.90(90.4/100)	0.06(298.3/4623.9), 0.75(298.3/400)	0.12(566.5/4623.9), 0.63(566.5/900)	0.18(822.6/4623.9), 0.51(822.6/1600)
Statlog	98.24 ± 1.10 (28878.2)	98.21 ± 1.12 (5647.8)	**SPSO**	$97.85^{-*} \pm 1.11$ (100)	98.12 ± 1.06 (400)	$98.11^{-*} \pm 1.19$ (900)	98.17 ± 1.10 (1600)
			CFA-PSO	$97.83^{-*} \pm 1.12$ (90.3)	$97.99^{-*} \pm 1.20$ (302.2)	$98.02^{-*} \pm 1.13$ (567.7)	$98.09^{-*} \pm 1.21$ (870.1)
			Ratio	0.02(90.3/5647.8), 0.90(90.3/100)	0.05(302.2/5647.8), 0.76(302.2/400)	0.10(567.7/5647.8), 0.63(567.7/900)	0.15(870.1/5647.8), 0.54(870.1/1600)
Svmguide1	95.78 ± 0.53 (29631.8)	95.72 ± 0.55 (6121.3)	**SPSO**	$95.46^{-*} \pm 0.57$ (100)	95.57 ± 0.76 (400)	95.65 ± 0.61 (900)	95.69 ± 0.52 (1600)
			CFA-PSO	$95.51^{-*} \pm 0.56$ (88.8)	$95.60^{-*} \pm 0.64$ (293.6)	95.62 ± 0.64 (553.1)	95.73 ± 0.61 (857.3)
			Ratio	0.01(88.8/6121.3), 0.89(88.8/100)	0.05(293.6/6121.3), 0.73(293.6/400)	0.09(553.1/6121.3), 0.61(553.1/900)	0.14(857.3/6121.3), 0.54(857.3/1600)
Wine Quality	79.49 ± 0.83 (28318.4)	79.43 ± 0.83 (5169.0)	**SPSO**	79.34 ± 0.95 (100)	$79.29^{-*} \pm 0.83$ (400)	79.34 ± 0.82 (900)	79.46 ± 0.89 (1600)
			CFA-PSO	$79.20^{-*} \pm 0.90$ (89.0)	$79.29^{-*} \pm 0.84$ (294.6)	79.43 ± 0.86 (565.3)	79.48 ± 0.87 (826.1)
			Ratio	0.02(89.0/5169.0), 0.89(89.0/100)	0.06(294.6/5169.0), 0.74(294.6/400)	0.11(565.3/5169.0), 0.63(565.3/900)	0.16(826.1/5169.0), 0.52(826.1/1600)

PSO-40 is significantly better than SEB for one dataset and significantly worse then SEB for two datasets.

All tables indicate that the prediction accuracy of CFA-PSO-30 and CFA-PSO-40 are equivalent to that of SPSO-30 and SPSO-40, respectively. In Table 5.2, CFA-PSO-30 is significantly worse than SPSO-30 for two datasets, and CFA-PSO-40 is significantly worse than SPSO-40 for four datasets. In Table 5.3, CFA-PSO-30 nor CFA-PSO-40 is not significantly different from SPSO-30 and SPSO-40 respectively for all datasets. In Table 5.4, CFA-PSO-30 is not significantly different from SPSO-30 respectively for all datasets, CFA-PSO-40 is significantly better than SPSO-40 for one dataset.

With respect to efficiency, the performance of PSO is generally high and CFA-PSO can save the decent amount of computation of PSO. This can be seen by looking at Ratio (the ratios of the number of SVM runs for CFA-PSO to that for SEB and to that for SPSO) reported in Tables 5.2, 5.3 and 5.4. It can be seen that as the number of particles and generations increase, the ratio decreases; that is, more computation is saved. This is because the benefit of a cache becomes observable when the overhead of saving fitness values in the cache is offset by a large number of reads from the cache.

In Tables 5.2 - 5.4, we report the number of SVM runs for each method and also report the ratio of the average number of SVM runs for CFA-PSO to that for SEB and to that for SPSO in the "Ratio" column. In Table 5.2, the average Ratio for CFA-PSO-30 to SPSO-30 for all datasets is 0.20 and the average Ratio for CFA-PSO-40 to SPSO-40 is 0.14. The average Ratio for CFA-PSO-30 to SEB for all datasets is 0.71 and the average Ratio for CFA-PSO-40 to SEB is 0.91. In Table 5.3, the average Ratio for CFA-PSO-30 to SPSO-30 for all datasets is 0.43 and the average Ratio for CFA-PSO-40 to SPSO-40 is 0.33. The average Ratio for CFA-PSO-30 to SEB for all datasets is 0.51 and the average Ratio for CFA-PSO-40 to SEB is 0.70. In Table 5.4, the average Ratio for CFA-PSO-30 to SPSO-30 for all datasets is 0.62 and the average Ratio for CFA-PSO-40 to SPSO-40 is 0.53. The average Ratio for CFA-PSO-30 to SEB for all datasets is 0.10 and the average Ratio for CFA-PSO-40 to SEB is 0.15. Note that we use the polynomial kernel with $b = 1$ for three-dimensional experiment in which the degree is integer, there is not so a large computational saving in Table 5.3 compared to the four dimensional experiment. For the four-dimensional experiment, CFA-PSO-30 has achieved an equivalent prediction accuracy when compared to SEB, with the saving of 90% of computation, and CFA-PSO-40 has achieved an equivalent prediction accuracy when compared to SEB, with the saving of 85% of computation without sacrificing accuracy.

Overhead of the cache process for recording marginal distribution on each axis is negligible. For the actual running times of the cache process for recording the history of position of particles, it takes an average of 0.01 seconds (and maximum of 0.02 seconds) for CFA-PSO-10, an average of 0.06 seconds (and maximum of 0.11 seconds) for CFA-PSO-20, an average of 0.23 seconds (and maximum of 0.33 seconds) for CFA-PSO-30, an average of 0.56 seconds (and maximum of 0.74 seconds) for CFA-PSO-40 for 17 datasets in four dimensional polynomial experiments. This time does not depend on the number of training examples or number of features as it is performed in the estimated hyper-parameter space.

For new datasets and new kernels, we would recommend setting large values for

the population sizes and the number of generations. In such settings, CFA-PSO can save unnecessary runs of the SVMs for the fitness evaluation while finding solutions that are at least as good as other methods.

5.4 Summary

We conducted two-, three-, and four-dimensional experiments to show that the discrete PSO with cellular fitness approximation can improve the efficiency of surface estimation methods without negatively impacting the accuracy. PSO also provides a means to automatically adjust the local search region. Therefore, PSO with cellular fitness approximation is more efficient and automatic than the surface estimation methods (or grid search) especially when the dimension of the search space is greater than two.

Contributions and achievements:

- In order to enhance the efficiency of the PSO, we proposed a new discrete PSO (based on the previous chapter) which dynamically changes the calibration between evaluation points according to the density of particles.
- Part of this efficiency improvement is achieved by proposing two types of cache processes: one for recording marginal distribution on each axis and the other one for recording the history of position of particles.
- Through the experiments in two-, three- and four-dimensional spaces, we showed that the proposed discrete PSO with the dynamic calibration improves the efficiency of the Bézier curve methods (SEB) without sacrificing accuracy.

Part III

Optimal Kernel Construction in SVMs

Chapter 6

Weight-Radius Multiple Kernel Learning

Generalization error rates of support vector machines are related to the ratio of radius of sphere which includes all the data and the margin between the separating hyperplane and the data as shown in Theorem 2.3. Therefore it would be rather natural to incorporate those quantities into the formulation of MKL (SVM). In this chapter we show that it is indeed possible. We prove that the proposed MKL is closed and has a global optimal solution.

6.1 Introduction

In (Vapnik, [160]), it was shown that the generalization error rates for linear hyperplane models are closely related to the ratio of the radius of enclosing spheres of data and the margin. Later this theorem was extended to nonlinear (kernel-based) hyperplane models (Bartlett and Shawe-Taylor, [5]). In this chapter we propose a combination of standard formulations of support vector machines (SVMs) and support vector data descriptions (SVDDs) for multiple kernel learning (MKL) to directly minimize the product of norms of weights of hyperplanes and radius, which we call the WR-MKL models. The way in which we combine SVM and SVDD guarantees finding the optimal coefficients of kernels without imposing additional constraints such as normalization of kernels or ℓ_p-norm of the coefficients of kernels.

In the literature, several authors state the importance of the ratio of radius and margin (the reciprocal of the norm of weights) and applied it to MKL (Do et al. [35] [36]) (Liu et al. [96] [97]). In (Gai et al. [49]), the authors firstly proposed the application

of the combination of SVMs and SVDDs to multiple kernel learning and formulated a bi-level optimization problem. We take the latter SVM-SVDD approach and improve it in several ways. Firstly we show that the proposed model can be formulated as a standard one-level optimization problem. Then we prove the existence of global optimal solutions and derive the dual feasible objective values which are used as a stopping criteria. Lastly we conduct experiments to show the good prediction accuracy of the proposed method by comparing the performance with the ℓ_p-norm MKL (Kloft et al. [77]) with fine tuned parameters.

A motivation of the research is to attain a better prediction accuracy (generalization error rate) by minimizing the ratio of radius and margin rather than just maximizing margin. Another motivation is related to the scale problem of SVM. Suppose that we have obtained an optimal hyperplane by which data is separated with a margin γ using a training set $\mathcal{S} = (\mathbf{x}, y_1), (\mathbf{y}, y_2), \ldots, (\mathbf{x}_n, y_n)$. If we use an another training set $\mathcal{S}' = (\alpha\mathbf{x}, y_1), (\alpha\mathbf{y}, y_2), \ldots, (\alpha\mathbf{x}_n, y_n)$ which is obtained by multiplying the input vector in $\mathcal{S}$ by a positive constant α, the data is separated by the hyperplane with a margin $\alpha\gamma$ (Shalev-Shwartz and Ben-David [137]). Therefore we always need to implicitly take care of the scale of data when we use SVMs. However, by incorporating a radius into the formulation, the objective values of SVMs become invariant for the multiplication of the input vector by a positive scaler α.

Gap or weakness in previous research: The idea of using the ratio of margin and radius instead of just margin for the construction of MKL has been explored by a number of authors. However, these methods have some problems:

- It has not been known whether the system has a unique global optimal solution.
- It has not been known whether the system is closed without additional constraints. Closeness is essential in order to understand and improve the current system.
- The problem has been formulated as a bi-level optimization problem and therefore requires solving nested optimization problems.
- There is no proof for the convergence (of any of the algorithms) and there is no clear convergence criteria either.

Objectives of Chapter 6: The goal is to devise a new MKL model which is suitable for kernel construction. More specifically the goal is to formulate a new WR-MKL as a

standard one-level optimization problem, to prove the existence of the global solution, and to derive a method to numerically find the optimal solution.

6.2 Related Work

In this section, we review the basic concepts of MKL and recent works in the literature.

6.2.1 Multiple Kernel Learning

The basic idea of *multiple kernel learning* (MKL) is to use a linear combination of kernels

$$K_{\boldsymbol{\theta}} = \theta_1 K_1 + \cdots + \theta_m K_m \tag{6.1}$$

in the dual form. To achieve this, the inner product $\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle$ is replaced by the sum $\sum_{k=1}^{m} \langle \mathbf{w}_k, \sqrt{\theta_k} \Phi_k(\mathbf{x}_i) \rangle$ in the primal form. However, to avoid the non-convexity arising from $\sqrt{\theta_k}$ we substitute $\sqrt{\theta_k}\mathbf{w}_k$ by $\hat{\mathbf{w}}_k$. For the binary classification problem in (2.7), the MKL optimization problem is formulated as follows:

$$\begin{aligned} \min_{\hat{\mathbf{w}}, b, \boldsymbol{\xi}, \boldsymbol{\theta}: \boldsymbol{\theta} \geq 0} & \frac{\lambda}{2} \sum_{k=1}^{m} \frac{\|\hat{\mathbf{w}}_k\|^2}{\theta_k} + \sum_{i=1}^{n} \xi_i \\ \text{s.t. } & y_i \Big(\sum_{k=1}^{m} \langle \hat{\mathbf{w}}_k, \Phi_k(\mathbf{x}_i) \rangle + b \Big) \geq 1 - \xi_i, \qquad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \ . \end{aligned} \tag{6.2}$$

Here we use a positive constant λ instead of C in (2.7).

Let $\|\mathbf{w}\|_p$ denote the p-norm of the parameters $\mathbf{w}$. The p-norm is defined as follows:

$$\|\mathbf{w}\|_p = \left(\sum_{i=1}^{d} |w_i|^p \right)^{\frac{1}{p}} .$$

With the following additional constraint

$$\|\boldsymbol{\theta}\|_p^2 \leq 1, \quad p \geq 1 \ ,$$

the optimization problem (6.2) can be solved with respect to $(\hat{\mathbf{w}}, b, \boldsymbol{\xi}, \boldsymbol{\theta})$. That is

$$\min_{\hat{\mathbf{w}},b,\boldsymbol{\xi},\boldsymbol{\theta}:\boldsymbol{\theta}\geq 0} \frac{\lambda}{2}\sum_{k=1}^{m}\frac{\|\hat{\mathbf{w}}_k\|^2}{\theta_k}+\sum_{i=1}^{n}\xi_i$$
$$\text{s.t. } y_i(\sum_{k=1}^{m}\langle\hat{\mathbf{w}}_k,\Phi_k(\mathbf{x}_i)\rangle+b)\geq 1-\xi_i, \qquad i=1,\ldots,n \tag{6.3}$$
$$\xi_i\geq 0, \quad i=1,\ldots,n$$
$$\|\boldsymbol{\theta}\|_p^2\leq 1 \ .$$

This is known as the ℓ_p-norm MKL. When $p=1$, ℓ_p-norm MKL produces sparse solutions. That is, $\theta_k=0$ for most k. Therefore we can control the sparsity of solutions using different values of k. ℓ_p-norm MKL is solved in the dual form using a coordinate descent method with respect to $(\hat{\mathbf{w}},b,\boldsymbol{\xi})$ and $\boldsymbol{\theta}$ (Kloft et al. [77]).

Using the following transformation (Micchelli and Pontil [107]), ℓ_p-norm MKL can be solved in the primal form too. For $r>0, p=1+\frac{1}{r}$, we set $q=r+1$, so that $\frac{1}{p}+\frac{1}{q}=1$. With the constraint $\sum_{k=1}^{m}\theta_k^r\leq 1$,

$$\begin{aligned}\sum_{k=1}^{m}|\hat{\mathbf{w}}_k|^{\frac{2}{p}}=\sum_{k=1}^{m}|\hat{\mathbf{w}}_k|^{\frac{2r}{q}} &= \sum_{k=1}^{m}\frac{|\hat{\mathbf{w}}_k|^{\frac{2r}{q}}}{\theta_k^{\frac{r}{q}}}\theta_k^{\frac{r}{q}}\\ &\leq \left(\sum_{k=1}^{m}\frac{|\hat{\mathbf{w}}_k|^{\frac{2rp}{q}}}{\theta_k^{\frac{rp}{q}}}\right)^{\frac{1}{p}}\left(\sum_{k=1}^{m}\theta_k^r\right)^{\frac{1}{q}}\\ &= \left(\sum_{k=1}^{m}\frac{|\hat{\mathbf{w}}_k|^2}{\theta_k}\right)^{\frac{1}{p}}\left(\sum_{k=1}^{m}\theta_k^r\right)^{\frac{1}{q}}\\ &\leq \left(\sum_{k=1}^{m}\frac{|\hat{\mathbf{w}}_k|^2}{\theta_k}\right)^{\frac{1}{p}} \ .\end{aligned}$$

The inequality in the second row follows from the Hölder inequality. Therefore, $\left(\sum_{k=1}^{m}|\hat{\mathbf{w}}_k|^{\frac{2}{p}}\right)^p\leq \sum_{k=1}^{m}\frac{|\hat{\mathbf{w}}_k|^2}{\theta_k}$, and the equality holds at the point $\theta_k=\frac{|\hat{\mathbf{w}}_k|^{\frac{2}{r+1}}}{\left(\sum_{k=1}^{m}|\hat{\mathbf{w}}_k|^{\frac{2r}{r+1}}\right)^{\frac{1}{r}}}$. Let $p'=\frac{2}{p}$. Then,

$$\left(\sum_{k=1}^{m}|\hat{\mathbf{w}}_k|^{\frac{2}{p}}\right)^p=\left(\sum_{k=1}^{m}|\hat{\mathbf{w}}_k|^{p'}\right)^{\frac{2}{p'}}$$

When $r\geq 1$, $1\leq p'\leq 2$.

The $(2,p)$ group norm $\|\mathbf{w}\|_{2,p}$ of $\mathbf{w}$ is defined as follows:

$$\|\mathbf{w}\|_{2,p}=\|(\|\mathbf{w}_1\|_2,\ldots,\|\mathbf{w}_m\|_2)\|_p\,.$$

With the condition $1 \leq p \leq 2$ and using the hinge loss function in (2.9), the ℓ_p-norm MKL can be equivalently formulated as follows:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{\lambda}{2} \|\mathbf{w}\|_{2,p}^2 + \sum_{i=1}^{n} \sum_{i=1}^{n} \ell\left(\mathbf{w}, \Phi(\mathbf{x}_i), y_i\right) \quad , \tag{6.4}$$

where λ is a regularization hyper-parameter. Jie et al. [99] applied the "Follow the Regularized Leader" method in online learning (Shalev-Shwartz [135], [136]) and solved (6.4) with the generalized hinge loss function given in (2.10).

Orabona et al. [117] proposed the following MKL method with an elastic net form of regularization function in order to more rigorously control the sparsity of the solutions:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{\lambda}{2} \|\mathbf{w}\|_{2,p}^2 + \alpha \|\mathbf{w}\|_{2,1} + \frac{1}{n} \sum_{i=1}^{n} \sum_{i=1}^{n} \ell\left(\mathbf{w}, \Phi(\mathbf{x}_i), y_i\right) \quad , \tag{6.5}$$

where p is set to $\frac{2 \log m}{2 \log m - 1}$. It is called the UFO-MKL, and it is also solved by applying the "Follow the Regularized Leader" method. In this model, α controls the sparsity of the solutions.

6.2.2 Recent Works of MKL

Kernels which are inner products of nonlinear mappings multiply the expressive power of SVMs. However, they scale poorly with the size of data, since the number of rows (columns) is equal to the number of instances of data. There have been many attempts to scale up the kernel methods.

Nyström method approximates $n \times n$ matrix K with rank-k ($k < n$) matrix (Williams and Seeger [168]). Using Singular Value Decomposition (SVD), the best rank-k approximation of K is computed as $K \sim U_k \Sigma_k U_k^T$ where Σ_k is a diagonal matrix with the largest singular values and U_k consists of the corresponding singular vectors. However the SVD of K is computationally prohibitive for large data. Let C be a $n \times m$ matrix by randomly selecting m columns from K, and let M be a $m \times m$ matrix between the m sample points. The Nyström method constructs rank-k approximation to K as follows:

$$K \sim C M_k^+ C^T$$

where M_k is the best rank-k approximation to M by SVD, and M_k^+ is its pseudo-inverse.

The random feature method proposed by (Rahimi and Recht [124]) approximates the kernel function instead of the kernel matrix. A shift-invariant kernel ($k(\mathbf{x}, \mathbf{y}) =$

$\kappa(\mathbf{x}-\mathbf{y})$ in Section 2.3.3) has the following representation.

$$\begin{aligned}\kappa(\mathbf{x}-\mathbf{y}) &= \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^T(\mathbf{x}-\mathbf{y})} d\boldsymbol{\omega} \\ &= \mathbb{E}_{\boldsymbol{\omega}}\left[e^{i\boldsymbol{\omega}^T\mathbf{x}} e^{-i\boldsymbol{\omega}^T\mathbf{y}}\right] \end{aligned} \tag{6.6}$$

where $p(\boldsymbol{\omega})$ is a probability density function obtained by the Fourier transform of $\kappa(\Delta\mathbf{x})$.

$$p(\boldsymbol{\omega}) = \left(\frac{1}{2\pi}\right)^d \int e^{-i\boldsymbol{\omega}^T(\Delta\mathbf{x})} \kappa(\Delta\mathbf{x}) d(\Delta\mathbf{x})$$

For Gaussian kernel $k(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$, $p(\boldsymbol{\omega})$ is a probability density of the normal distribution with mean $\mathbf{0}$ and variance $\sigma^{-2}I$:

$$p(\boldsymbol{\omega}) = \mathcal{N}(\mathbf{0}, \sigma^{-2}I). \tag{6.7}$$

In (6.6),

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\omega}}\left[e^{i\boldsymbol{\omega}^T\mathbf{x}} e^{-i\boldsymbol{\omega}^T\mathbf{y}}\right] &= \mathbb{E}_{\boldsymbol{\omega}}\left[\cos(\boldsymbol{\omega}^T\mathbf{x})\cos(\boldsymbol{\omega}^T\mathbf{y}) + \sin(\boldsymbol{\omega}^T\mathbf{x})\sin(\boldsymbol{\omega}^T\mathbf{y})\right] \\ &= \mathbb{E}_{\boldsymbol{\omega}}\left[\left(\cos(\boldsymbol{\omega}^T\mathbf{x}), \sin(\boldsymbol{\omega}^T\mathbf{x})\right), \left(\cos(\boldsymbol{\omega}^T\mathbf{y}), \sin(\boldsymbol{\omega}^T\mathbf{y})\right)\right] \end{aligned} \tag{6.8}$$

By drawing i.i.d sample $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_n$ from the probability density p in (6.7), The expectation in (6.8) is approximated by the sample mean $\frac{1}{n}\mathbf{z}(\mathbf{x})^T\mathbf{z}(\mathbf{y})$, where

$$\mathbf{z}(\mathbf{x}) = \left((\cos(\boldsymbol{\omega}_1^T\mathbf{x}), \sin(\boldsymbol{\omega}_1^T\mathbf{x})), \ldots, (\cos(\boldsymbol{\omega}_n^T\mathbf{x}), \sin(\boldsymbol{\omega}_n^T\mathbf{x}))\right).$$

Based on these two approximation methods, Lu et al. [98] construct online kernel learning methods for large-scale data.

Si et al. [141] observed that the structure of shift-invariant kernels changes from low-rank matrix to block-diagonal matrix by changing the value of the parameter. For instance, in the case of Gaussian kernel $k(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$, if $\sigma^2 \to \infty$, $K \to \mathbf{e}^T\mathbf{e}$ where $\mathbf{e} = (1, \ldots, 1)$. Therefore K is close to row-rank. If $\sigma^2 \to 0$, K approaches an identity matrix which does not have the row-rank property. Based on this observation, the authors propose the following kernel construction procedure. Firstly, the clustering structure of the kernel K is examined by applying k-means clustering method. Then, K is reordered according to the clustering. Finally, the low-rank approximation is performed on the diagonal blocks of K. Their experiments show the improved efficiency compared to the standard approximation methods such as the Nyström method.

6.2.3 Optimality Conditions and Duality for Generalized Convex Functions

In this section, we extend the optimality conditions and duality forms in Section 2.3.2 to generalized (weaker) convex functions. We start with the definition of quasiconvex and pseudoconvex functions. We restate the general form of the optimization problem (2.13) for convenience:

$$\begin{aligned} P: \quad & \min_{\mathbf{x}} \quad f(\mathbf{x}) \\ & \text{s.t.} \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \end{aligned}$$

Suppose that $\mathcal{C}$ is a convex set in $\mathbb{R}^d$. Then, a function $f(\mathbf{x})$ defined on C is quasiconvex or quasiconcave according to the following definition.

Definition 6.1. ***Quasiconvex*** *and* ***Quasiconcave*** *Functions (Mangasarian [103])*
A function $f(\mathbf{x})$ is quasiconvex on $\mathcal{C}$ if

$$\begin{cases} \mathbf{x}, \mathbf{y} \in \mathcal{C} \\ f(\mathbf{y}) \leq f(\mathbf{x}) \quad \Rightarrow \quad f\left[(1-\tau)\mathbf{x} + \tau\mathbf{y}\right] \leq f(\mathbf{x}) \\ 0 \leq \tau \leq 1 \end{cases} .$$

A function $f(\mathbf{x})$ is quasiconcave on $\mathcal{C}$ if

$$\begin{cases} \mathbf{x}, \mathbf{y} \in \mathcal{C} \\ f(\mathbf{x}) \leq f(\mathbf{y}) \quad \Rightarrow \quad f(\mathbf{x}) \leq f\left[(1-\tau)\mathbf{x} + \tau\mathbf{y}\right] \\ 0 \leq \tau \leq 1 \end{cases} .$$

Suppose that $f(\mathbf{x})$ is differentiable on an open set in $\mathbb{R}^d$ which contains the convex set $\mathcal{C}$.

Lemma 6.1. *(Cambini and Martein [22])*
The function $f(\mathbf{x})$ is quasiconvex on $\mathcal{C}$ if and only if the following implication holds:

$$f(\mathbf{y}) \leq f(\mathbf{x}) \Rightarrow \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \leq 0 \ ,$$

and $f(\mathbf{x})$ is quasiconcave on $\mathcal{C}$ if and only if the following implication holds:

$$f(\mathbf{x}) \leq f(\mathbf{y}) \Rightarrow \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \geq 0 \ .$$

For differentiable functions, pseudo-convexity and pseudo-concavity are defined as follows.

Definition 6.2. ***Pseudoconvex*** *and* ***Pseudoconcave*** *(Mangasarian [103])*
A function $f(\mathbf{x})$ is pseudoconvex on $\mathcal{C}$ if

$$f(\mathbf{y}) < f(\mathbf{x}) \Rightarrow \nabla f(\mathbf{x})(\mathbf{y}-\mathbf{x}) < 0$$
$$(\nabla f(\mathbf{x})(\mathbf{y}-\mathbf{x}) \geq 0 \Rightarrow f(\mathbf{y}) \geq f(\mathbf{x}))\,.$$

A function $f(\mathbf{x})$ is pseudoconcave on $\mathcal{C}$ if

$$f(\mathbf{x}) < f(\mathbf{y}) \Rightarrow \nabla f(\mathbf{x})(\mathbf{y}-\mathbf{x}) > 0$$
$$(\nabla f(\mathbf{x})(\mathbf{y}-\mathbf{x}) \leq 0 \Rightarrow f(\mathbf{y}) \leq f(\mathbf{x}))\,.$$

Pseudoconvex functions have the following crucial property.

Lemma 6.2. *(Cambini and Martein [22])*
Let $\mathbf{x}_0 \in \mathcal{C}$ be a critical point (a point such that $\nabla f(\mathbf{x}) = 0$). If $f(\mathbf{x})$ is pseudoconvex, then $\mathbf{x}_0$ is a global minimum for $f(\mathbf{x})$.

If $f(\mathbf{x})$ is a pseudoconvex function, and the constraint functions $g_1(\mathbf{x}), \ldots, g_m(\mathbf{x})$ are quasiconvex, and if a constraint qualification such as Slater's constraint qualification (2.14) is satisfied, then the KKT conditions

$$g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \tag{6.9}$$
$$\lambda_i g_i(\mathbf{x}) = 0, \quad i = 1, \ldots, m \tag{6.10}$$
$$\lambda_i \geq 0, \quad i = 1, \ldots, m \tag{6.11}$$
$$\nabla \left(f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) \right) = 0 \tag{6.12}$$

are necessary and sufficient for the optimality of the solution (Mangasarian [103]).

For generalized convex functions, two types of dual forms are commonly used, the Wolfe dual and the Mond-Weir dual (Mond and Weir [111]). We state the definition of the Wolfe dual again and Lemma 2.3 in a more general form.

Wolfe dual: For the optimization problem P in (2.13), the Wolfe dual is defined as:

$$\begin{aligned} WD: \qquad \max_{\mathbf{u}} \quad & f(\mathbf{u}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{u}) \\ \text{s.t.} \quad & \nabla f(\mathbf{u}) + \nabla \sum_{i=1}^{m} \lambda_i g_i(\mathbf{u}) = 0 \\ & \lambda_i \geq 0, \quad i = 1, \ldots, m. \end{aligned}$$

Lemma 6.3. *(Bector et al. [9], Mond [110])*
If $f + \sum_{i=1}^{m} \lambda_i g_i$, for $\lambda_i \geq 0$, is pseudoconvex, the weak duality holds. If the optimization problem P has an optimal solution $\mathbf{x} = \mathbf{x}_0$ and a constraint qualification is satisfied, then there exists a $\boldsymbol{\lambda}_0$, such that $(\mathbf{u} = \mathbf{x}_0, \boldsymbol{\lambda}_0)$ is optimal for WD, and strong duality holds.

Mond-Weir dual: For the optimization problem P in (2.13), the Mond-Weir dual is defined as:

$$\begin{aligned} MWD: \quad \max_{\mathbf{u}} \quad & f(\mathbf{u}) \\ \text{s.t.} \quad & \nabla f(\mathbf{u}) + \nabla \sum_{i=1}^{m} \lambda_i g_i(\mathbf{u}) = 0 \\ & \sum_{i=1}^{m} \lambda_i g_i(\mathbf{u}) \geq 0 \\ & \lambda_i \geq 0, \quad i = 1, \ldots, m. \end{aligned} \tag{6.13}$$

Lemma 6.4. *(Mond and Weir [111])*
If f is pseudoconvex and $\sum_{i=1}^{m} \lambda_i g_i$, for $\lambda_i \geq 0$, is quasiconvex, the weak duality holds. If the optimization problem P has an optimal solution $\mathbf{x} = \mathbf{x}_0$ and a constraint qualification is satisfied, then there exists a $\boldsymbol{\lambda}_0$, such that $(\mathbf{u} = \mathbf{x}_0, \boldsymbol{\lambda}_0)$ is optimal for MWD, and strong duality holds.

Furthermore, the combination of those two dualities is also valid. Let $M = \{1, \ldots, m\}$ and S be a subset of M. If $f + \sum_{i \in S} \lambda_i g_i$, for $\lambda_i \geq 0$, is pseudoconvex and $\sum_{i \in M \setminus S} \lambda_i g_i$, for $\lambda_i \geq 0$, is quasiconvex, the weak duality and the strong duality hold for the following dual form (Mond and Weir [111]).

Combination of Wolfe and Mond-Weir dual:

$$\begin{aligned} W - MWD: \quad \max_{\mathbf{u}} \quad & f(\mathbf{u}) + \sum_{i \in S} \lambda_i g_i(\mathbf{u}) \\ \text{s.t.} \quad & \nabla f(\mathbf{u}) + \nabla \sum_{i=1}^{m} \lambda_i g_i(\mathbf{u}) = 0 \\ & \sum_{i \in M \setminus S} \lambda_i g_i(\mathbf{u}) \geq 0 \\ & \lambda_i \geq 0, \quad i = 1, \ldots, m. \end{aligned} \tag{6.14}$$

6.3 The Proposed WR-MKL Models

In this section we start with the multiple kernel learning (MKL) formulation of SVMs and SVDDs and then formulate our WR-MKL models. We also examine the convergence property of the proposed models.

6.3.1 MKL Formulation for SVMs

As stated in Section 6.2.1, usually the combinations of kernels are modeled as in (6.1):

$$K_{\boldsymbol{\theta}} = \theta_1 K_1 + \cdots + \theta_m K_m.$$

We use a simpler form of a linear combination of kernels:

$$K_{\boldsymbol{\theta}} = \theta_1^2 K_1 + \cdots + \theta_m^2 K_m \tag{6.15}$$

to avoid the denominator of θ_k in $\sum_{k=1}^{m} \frac{\|\hat{\mathbf{w}}_k\|^2}{\theta_k}$ in (6.2) .

We formulate the MKL optimization problem for binary classification of SVM as follows:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\theta}} & \frac{\lambda}{2}\sum_{k=1}^{m}\|\mathbf{w}_k\|^2 + \sum_{i=1}^{n}\xi_i \\ \text{s.t. } & y_i(\sum_{k=1}^{m}\langle \mathbf{w}_k, \theta_k\Phi_k(\mathbf{x}_i)\rangle + b) \geq 1-\xi_i, \qquad i=1,\ldots,n \\ & \xi_i \geq 0, \quad i=1,\ldots,n \end{aligned} \tag{6.16}$$

The Lagrangian L is:

$$\begin{aligned} L(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\theta}) &= \frac{\lambda}{2}\sum_{k=1}^{m}\|\mathbf{w}_k\|^2 + \sum_{i=1}^{n}\xi_i \\ &- \sum_{i=1}^{n}\alpha_i\left[y_i(\sum_{k=1}^{m}\langle \mathbf{w}_k, \theta_k\Phi_k(\mathbf{x}_i)\rangle + b) - 1 + \xi_i\right] - \sum_{i=1}^{n}\nu_i\xi_i \end{aligned}$$

where $\boldsymbol{\alpha} \geq 0$ and $\boldsymbol{\nu} \geq 0$ are Lagrange multipliers.
Setting the partial derivatives to zero gives the constraints:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}_k} = 0 &\Rightarrow \lambda\mathbf{w}_k - \sum_{i=1}^{n}\theta_k\alpha_i y_i\Phi_k(\mathbf{x}_i) = 0 \\ \frac{\partial L}{\partial b} = 0 &\Rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0 \\ \frac{\partial L}{\partial \xi_i} = 0 &\Rightarrow 1-\alpha_i-\nu_i = 0. \end{aligned}$$

The Wolfe dual of (6.16) for a fixed $\boldsymbol{\theta}$ is derived as:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda} \left(\boldsymbol{\alpha} \circ \mathbf{y}\right)^T K_\theta \left(\boldsymbol{\alpha} \circ \mathbf{y}\right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq 1, \quad i = 1, \ldots, n \end{aligned} \tag{6.17}$$

where $\boldsymbol{\alpha} \circ \mathbf{y}$ is an element-wise product between a vector $\boldsymbol{\alpha}$ and a vector $\mathbf{y}$ and $K_\theta = \sum_{k=1}^{m} \theta_k^2 K_k$.

6.3.2 MKL Formulation for SVDDs

In our formulation we use the "hard-margin" SVDDs with a positive constant λ:

$$\begin{aligned} & \min_{R, \mathbf{a}, \boldsymbol{\theta}} \frac{\lambda}{2} R \\ & \text{s.t.} \sum_{k=1}^{m} \theta_k^2 \left\|\Phi_k(\mathbf{x}_i)\right\|^2 - 2 \sum_{k=1}^{m} \langle \theta_k \Phi_k(\mathbf{x}_i), \mathbf{a}_k \rangle + \sum_{k=1}^{m} \left\|\mathbf{a}_k\right\|^2 \leq R \\ & \quad \text{for} \quad i = 1, \ldots, n. \end{aligned} \tag{6.18}$$

We construct the Lagrangian L as:

$$\begin{aligned} & L(R, \mathbf{a}, \boldsymbol{\beta}, \boldsymbol{\theta}) = \frac{\lambda}{2} R + \\ & \sum_{i=1}^{n} \beta_i \left(\sum_{k=1}^{m} \theta_k^2 \left\|\Phi_k(\mathbf{x}_i)\right\|^2 - 2 \sum_{k=1}^{m} \langle \theta_k \Phi_k(\mathbf{x}_i), \mathbf{a}_k \rangle + \sum_{k=1}^{m} \left\|\mathbf{a}_k\right\|^2 - R \right) \end{aligned} \tag{6.19}$$

where $\boldsymbol{\beta} \geq 0$ are Lagrange multipliers.

Setting the partial derivatives to zero we obtain the following constraints:

$$\frac{\partial L}{\partial R} = 0 \quad \Rightarrow \quad \frac{\lambda}{2} - \sum_{i=1}^{n} \beta_i = 0 \tag{6.20}$$

$$\frac{\partial L}{\partial \mathbf{a}_k} = 0 \quad \Rightarrow \quad - \sum_{i=1}^{n} \beta_i \theta_k \Phi_k(\mathbf{x}_i) + \sum_{i=1}^{n} \beta_i \mathbf{a}_k = 0 \tag{6.21}$$

Setting $\tilde{\beta}_i = \frac{\beta_i}{\sum_{i=1}^{n} \beta_i}$ and substituting those values into L , the Wolfe dual of (6.18) for a fixed $\boldsymbol{\theta}$ is computed as:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \max_{\tilde{\boldsymbol{\beta}}} \quad & \frac{\lambda}{2} \left(\sum_{i=1}^{n} \tilde{\beta}_i K_{\theta,ii} - \tilde{\boldsymbol{\beta}}^T K_\theta \tilde{\boldsymbol{\beta}} \right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} \tilde{\beta}_i = 1 \\ & 0 \leq \tilde{\beta}_i, \quad i = 1, \ldots, n. \end{aligned} \tag{6.22}$$

where $K_\theta = \sum_{k=1}^m \theta_k^2 K_k$ and $K_{\theta,ii}$ is the element of the i-th row and the i-th column of K_θ.

Lemma 6.5. *For the optimization problem (6.18), the optimal R is derived as:*

$$R = K_{\theta,ii} - 2K_{\theta,i\cdot}\tilde{\boldsymbol{\beta}} + \tilde{\boldsymbol{\beta}}^T K_\theta \tilde{\boldsymbol{\beta}} \tag{6.23}$$

for the instances $\mathbf{x}_i$ with $\tilde{\beta}_i > 0$ and the optimal R does not change for any values of λ.

Proof. The optimal R in (6.23) is derived from the KKT condition in (6.10) which requires that

$$\beta_i \left(\sum_{k=1}^m \theta_k^2 \left\| \Phi_k(\mathbf{x}_i) \right\|^2 - 2 \sum_{k=1}^m \langle \theta_k \Phi_k(\mathbf{x}_i), \mathbf{a}_k \rangle + \sum_{k=1}^m \left\| \mathbf{a}_k \right\|^2 - R \right) = 0 \quad i = 1, \ldots, n.$$

Since it does not depend on λ, the latter part of the claim follows. It is also obvious from the fact that for monotone increasing functions $\psi(\mathbf{x})$,

$$\mathrm{argmin}_{\mathbf{x} \in X} \psi(\mathbf{x}) = \mathrm{argmin}_{\mathbf{x} \in X} \lambda \psi(\mathbf{x})$$

for any $\lambda > 0$. □

6.3.3 Formulation of WR-MKL Models

Combining the two optimization problems (6.16) and (6.18), we formulate the WR-MKL Models as follows.

$$\begin{aligned} \min_{\boldsymbol{\theta}, \mathbf{w}, b, R, \mathbf{a}, \boldsymbol{\xi}} & \frac{\lambda}{2} \sum_{k=1}^m \left\| \mathbf{w}_k \right\|^2 R + \sum_{i=1}^n \xi_i \\ \text{s.t. } & y_i \Big(\sum_{k=1}^m \langle \mathbf{w}_k, \theta_k \Phi_k(\mathbf{x}_i) \rangle + b \Big) \geq 1 - \xi_i, \qquad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n. \\ & \sum_{k=1}^m \left\| \theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k \right\|^2 \leq R, \; i = 1, \ldots, n \end{aligned} \tag{6.24}$$

We assume that $\sum_{k=1}^m \left\| \mathbf{w}_k \right\|^2 > 0$ and $R > 0$.

6.3.4 Existence of a unique Global Optimal Solution

$\sum_{k=1}^m \left\| \theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k \right\|^2 \leq R$ for $i = 1, \ldots, n$ is equivalently formulated as

$$\max_i \sum_{k=1}^m \left\| \theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k \right\|^2 \leq R.$$

The minimal R which satisfies this condition is $R = \max_i \sum_{k=1}^{m} \|\theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k\|^2$. Therefore the optimization problem (6.24) is equivalently formulated as:

$$\min_{\boldsymbol{\theta},\mathbf{w},b,\mathbf{a}} \frac{\lambda}{2} \sum_{k=1}^{m} \|\mathbf{w}_k\|^2 \max_i \sum_{k=1}^{m} \|\theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k\|^2 + \sum_{i=1}^{n} \max\left(0, 1 - y_i(\sum_{k=1}^{m} \langle \mathbf{w}_k, \theta_k \Phi_k(\mathbf{x}_i)\rangle + b)\right) \tag{6.25}$$

in an unconstrained form. We show the existence of a unique global optimal solution using the formulation (6.25).

First of all we notice that (6.25) is not an convex problem. To see it let's consider a function $h(x, y) = x^2 y^2$. $h(1, 2) = h(2, 1) = 4$. At the midpoint between $(1, 2)$ and $(2, 1)$, $h(1.5, 1.5) = 5.0625$. Therefore it is not a convex function because it violates the condition (2.1). Therefore,

$$\sum_{k=1}^{m} \|\mathbf{w}_k\|^2 \max_i \sum_{k=1}^{m} \|\theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k\|^2$$

is not convex with respect to $(\boldsymbol{\theta}, \mathbf{w}, \mathbf{a})$.

However the problem (6.25) is convex for each $\boldsymbol{\theta}$, $\mathbf{w}$ and $\mathbf{a}$. We will show that it has a unique global optimal solution. Let us start with some definitions.

Definition 6.3. *Continuous Function*
Let X be a subset of $\mathbb{R}^d$. A function $f : X \mapsto \mathbb{R}$ is continuous at $\mathbf{x} \in X$, if $\lim_{\mathbf{z}\to\mathbf{x}} f(\mathbf{z}) = f(\mathbf{x})$.

Definition 6.4. *Semi-continuous Function*
Let X be a subset of $\mathbb{R}^d$. A function $f : X \mapsto \mathbb{R}$ is upper-semi continuous at $\mathbf{x} \in X$, if $f(\mathbf{x}) \geq \limsup_{k\to\infty} f(\mathbf{x}_k)$ and is lower-semi continuous if $f(\mathbf{x}) \leq \liminf_{k\to\infty} f(\mathbf{x}_k)$. An example of a semi continuous function is an indicator function on a set C:

$$\imath_C : X \mapsto [-\infty, +\infty] : \mathbf{x} = \begin{cases} 0, & \text{if } \mathbf{x} \in C \\ +\infty, & \text{otherwise.} \end{cases}$$

If C is an open set, the function $\imath_C$ is upper-semi continuous. If C is a closed set, the function $\imath_C$ is lower-semi continuous.

Definition 6.5. *Coercive*
A function $f : X \mapsto \mathbb{R}$ is called coercive, if for every sequence $\{\mathbf{x}_k\} \subset X$ such that $\|\mathbf{x}_k\| \to \infty$, $\lim_{k\to\infty} f(\mathbf{x}_k) = \infty$.

Definition 6.6. *Limit points (Bertsekas [11])*
A vector $\mathbf{x} \in \mathbb{R}^d$ is a limit point of a sequence $\{\mathbf{x}_k\} \in \mathbb{R}^d$, if there is a subsequence of $\{\mathbf{x}_k\}$ that converges to $\mathbf{x}$.

Let in the formulation (6.25):

$$
\begin{aligned}
g_1(\mathbf{w}) &= \frac{\lambda}{2}\sum_{k=1}^{m}\|\mathbf{w}_k\|^2 \\
g_2(\boldsymbol{\theta},\mathbf{a}) &= \max_i \sum_{k=1}^{m}\|\theta_k\Phi_k(\mathbf{x}_i)-\mathbf{a}_k\|^2 \\
g_3(\mathbf{w},\boldsymbol{\theta}) &= \sum_{i=1}^{n}\max\left(0, 1-y_i(\sum_{k=1}^{m}\langle \mathbf{w}_k,\theta_k\Phi_k(\mathbf{x}_i)\rangle + b)\right)
\end{aligned}
\tag{6.26}
$$

Continuous functions are lower-semi continuous and from the fact that the supreme (maximum) of lower-semi continuous functions is lower-semi continuous (Bauschke and Combettes [8]), $g_2(\boldsymbol{\theta},\mathbf{a})$ is lower-semi continuous with respect to $(\boldsymbol{\theta},\mathbf{a})$.

$$
\frac{\lambda}{2}\sum_{k=1}^{m}\|\mathbf{w}_k\|^2 \max_i \sum_{k=1}^{m}\|\theta_k\Phi_k(\mathbf{x}_i)-\mathbf{a}_k\|^2
$$

is a product of g_1 (which depends only on $\mathbf{w}$) and g_2 (which depends only on $(\boldsymbol{\theta},\mathbf{a})$).

$$
\begin{aligned}
&\frac{\lambda}{2}\sum_{k=1}^{m}\|\mathbf{w}_k^*\|^2 \max_i \sum_{k=1}^{m}\|\theta_k^*\Phi_k(\mathbf{x}_i)-\mathbf{a}_k^*\|^2 \\
=&g_1(\mathbf{w}^*)g_2(\boldsymbol{\theta}^*,\mathbf{a}^*) \\
\leq&\liminf_{\mathbf{w}\to\mathbf{w}^*} g_1(\mathbf{w}) \liminf_{(\boldsymbol{\theta},\mathbf{a})\to(\boldsymbol{\theta}^*,\mathbf{a}^*)} g_2(\boldsymbol{\theta},\mathbf{a}) \\
=&\liminf_{(\mathbf{w},\boldsymbol{\theta},\mathbf{a})\to(\mathbf{w}^*,\boldsymbol{\theta}^*,\mathbf{a}^*)} \frac{\lambda}{2}\sum_{k=1}^{m}\|\mathbf{w}_k\|^2 \max_i \sum_{k=1}^{m}\|\theta_k\Phi_k(\mathbf{x}_i)-\mathbf{a}_k\|^2
\end{aligned}
$$

Therefore, $g_1(\mathbf{w})g_2(\boldsymbol{\theta},\mathbf{a})$ is lower-semi continuous with respect to $(\mathbf{w},\boldsymbol{\theta},\mathbf{a})$. $g_3(\mathbf{w},\boldsymbol{\theta},\mathbf{a})$ is (sum of) the maximum of continuous functions, and so it is lower-semi continuous. Overall, the objective function in (6.25) is lower-semi continuous. The following theorem states the conditions for global optimal solutions.

Proposition 6.1. *Weierstrass Theorem (Bertsekas [11])*
Let $f : X \mapsto \mathbb{R}$ be lower-semi continuous on X. Assume that one of the following conditions holds:

1. *X is compact.*
2. *X is closed and f is coercive.*

Then, the set of optimal solutions f over X is nonempty and compact.

Since the objective function in (6.25) is lower-semi continuous and coercive with respect to $(\mathbf{w}, \boldsymbol{\theta}, \mathbf{a})$, it has optimal solutions. Since the objective function in (6.25) is strictly convex for each $\mathbf{w}$, $\boldsymbol{\theta}$ and $\mathbf{a}$, the compact set of optimal solutions should be a single point. Next we see the procedure to seek for the optimal point.

From Lemma 2.1, convex functions are sub-differential over $\mathcal{C}$. The optimal solutions for sub-differential functions $f(\mathbf{x})$ are characterized as

$$\mathbf{0} \in \partial f(\mathbf{x})$$

where $\partial f(\mathbf{x})$ is the sub-differential set of $f(\mathbf{x})$ as shown in the following Lemma.

Lemma 6.6. *Fermat Theorem (Bauschke [8])*
For a function $f : \mathcal{C} \mapsto \mathbb{R}$,

$$\operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}) = \{\mathbf{x} \in \mathcal{C} | \mathbf{0} \in \partial f(\mathbf{x})\} \tag{6.27}$$

Proof.

$$\mathbf{x} \in \operatorname{argmin} f \Leftrightarrow \forall \mathbf{y} \in \mathcal{C}, \langle \mathbf{y} - \mathbf{x}, \mathbf{0} \rangle + f(\mathbf{x}) \leq f(\mathbf{y}) \Leftrightarrow \mathbf{0} \in \partial f(\mathbf{x})$$

□

We call a vector $\mathbf{x}$ which satisfies (6.27) a "stationary point".

Block coordinate descent (Gauss-Seidel) methods are applied to solve the problem:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &\text{s.t. } \mathbf{x} \in X \end{aligned} \tag{6.28}$$

where X is a Cartesian product of closed convex sets $X_i \subset \mathbb{R}^{d_i}$ such that

$$X = X_1 \times \cdots \times X_l$$

and $d = d_1 + \cdots + d_l$. A vector $\mathbf{x}$ consists of block components

$$\mathbf{x} = (\mathbf{x}, \ldots, \mathbf{x}_l)$$

where $\mathbf{x}_i \in \mathbb{R}^{d_i}$. Assume that for every $\mathbf{x} \in X$ and every $i = 1, \ldots, l$, the optimization problem:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}, \ldots, \mathbf{x}_{i-1}, \boldsymbol{\zeta}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_l) \\ &\text{s.t. } \boldsymbol{\zeta} \in X_i \end{aligned}$$

has a solution. The block coordinate descent methods produce the solutions in the next iteration $\mathbf{x}^{k+1} = (\mathbf{x}^{k+1}, \ldots, \mathbf{x}_l^{k+1})$ based on the solutions in the current iteration $\mathbf{x}^k = (\mathbf{x}^k, \ldots, \mathbf{x}_l^k)$ by solving each $\mathbf{x}_i$ in a cyclic manner:

$$\mathbf{x}_i^{k+1} = \text{argmin}_{\boldsymbol{\zeta} \in X_i} f(\mathbf{x}^{k+1}, \ldots, \mathbf{x}_{i-1}^{k+1}, \boldsymbol{\zeta}, \mathbf{x}_{i+1}^k, \ldots, \mathbf{x}_l^k)$$

(Bertsekas [11]). The following theorem shows that the block coordinate descent methods converge to a stationary point.

Proposition 6.2. *Proposition 2.7.1 in (Bertsekas [11])*
Suppose that f in (6.28) is sub-differentiable for each $\mathbf{x}_i \in \mathbb{R}^{d_i}$. Suppose that for each $\mathbf{x} = (\mathbf{x}, \ldots, \mathbf{x}_l) \in X$ and each $i = 1, \ldots, l$

$$f(\mathbf{x}, \ldots, \mathbf{x}_{i-1}, \boldsymbol{\zeta}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_l)$$

attains a unique minimum $\bar{\boldsymbol{\zeta}}$ over X_i and monotonically non-increasing in the interval from $\mathbf{x}_i$ to $\bar{\boldsymbol{\zeta}}$. Let $\{\mathbf{x}_k\}$ be a sequence generated by the block coordinate descent method. Then, every limit point of $\{\mathbf{x}_k\}$ is a stationary point.

In Proposition 2.7.1 in (Bertsekas [11]), f is assumed to be continuously differentiable. Under the assumption that f is sub-differentiable for each $\mathbf{x}_i \in \mathbb{R}^{d_i}$, the proof is also valid without any changes except that the condition of the stationary point is changed to $\mathbf{0} \in \partial_i f(\mathbf{x}_i)$ for $\mathbf{x}_i$. Since the objective function in (6.25) is strictly convex for each $\mathbf{w}$, $\boldsymbol{\theta}$ and $\mathbf{a}$, the conditions in the above proposition are satisfied.

6.3.5 Algorithms

We apply the block coordinate descent methods to solve the optimization problem (6.24).

Step 1: optimization for $(R, \mathbf{a})$ for fixed $\boldsymbol{\theta}$ and $(\mathbf{w}, b, \boldsymbol{\xi})$:

From Lemma 6.5, for fixed $\boldsymbol{\theta}$ and $\mathbf{w}$ the optimal R and $\mathbf{a}$ in (6.24) can be solved using the optimization problem (6.22) with $\lambda = 1$.

Step 2: optimization for $(\mathbf{w}, b, \boldsymbol{\xi})$ for fixed $\boldsymbol{\theta}$ and $(R, \mathbf{a})$:

For fixed $\boldsymbol{\theta}$ and $(R, \mathbf{a})$, the optimal $\mathbf{w}$ and b can be obtained using the optimization problem (6.17) with $\tilde{\lambda} = \lambda R$.

Step 3: optimization for $\boldsymbol{\theta}$ for fixed ($\mathbf{w}, b, \boldsymbol{\xi}$) and ($R, \mathbf{a}$):

We construct the Lagrangian L of (6.24) for binary classification as:

$$\begin{aligned} & L(\boldsymbol{\theta}, \mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\nu}, R, \mathbf{a}, \boldsymbol{\beta}) \\ = & \frac{\lambda}{2} \sum_{k=1}^{m} \|\mathbf{w}_k\|^2 R + \sum_{i=1}^{n} \xi_i \\ - & \sum_{i=1}^{n} \alpha_i \left[y_i \left(\sum_{k=1}^{m} \langle \mathbf{w}_k, \theta_k \Phi_k(\mathbf{x}_i) \rangle + b \right) - 1 + \xi_i \right] - \sum_{i=1}^{n} \nu_i \xi_i \\ + & \sum_{i=1}^{n} \beta_i \left[\sum_{k=1}^{m} \theta_k^2 \|\Phi_k(\mathbf{x}_i)\|^2 - 2 \sum_{k=1}^{m} \langle \theta_k \Phi_k(\mathbf{x}_i), \mathbf{a}_k \rangle + \sum_{k=1}^{m} \|\mathbf{a}_k\|^2 - R \right]. \end{aligned} \tag{6.29}$$

where $\boldsymbol{\alpha} \geq 0$ and $\boldsymbol{\beta} \geq 0$ are Lagrange multipliers.
Setting the partial derivatives with respect to R to zero gives the constraint:

$$\frac{\partial L}{\partial R} = 0 \Rightarrow \frac{\lambda}{2} \sum_{k=1}^{m} \|\mathbf{w}_k\|^2 = \sum_{i=1}^{n} \beta_i$$

Setting $\tilde{\beta}_i = \frac{\beta_i}{\sum_{i=1}^{n} \beta_i}$,

$$\begin{aligned} & L(\boldsymbol{\theta}, \mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\nu}, R, \mathbf{a}, \boldsymbol{\beta}) \\ = & \frac{\lambda}{2} \sum_{k=1}^{m} \|\mathbf{w}_k\|^2 R + \sum_{i=1}^{n} \xi_i \\ & - \sum_{i=1}^{n} \alpha_i \left[y_i \left(\sum_{k=1}^{m} \langle \mathbf{w}_k, \theta_k \Phi_k(\mathbf{x}_i) \rangle + b \right) - 1 + \xi_i \right] - \sum_{i=1}^{n} \nu_i \xi_i \\ & + \frac{\lambda}{2} \sum_{k=1}^{m} \|\mathbf{w}_k\|^2 \sum_{i=1}^{n} \tilde{\beta}_i \left[\sum_{k=1}^{m} \theta_k^2 \|\Phi_k(\mathbf{x}_i)\|^2 - 2 \sum_{k=1}^{m} \langle \theta_k \Phi_k(\mathbf{x}_i), \mathbf{a}_k \rangle + \sum_{k=1}^{m} \|\mathbf{a}_k\|^2 - R \right]. \end{aligned} \tag{6.30}$$

We can solve the optimal θ_k analytically.

$$\begin{aligned} & \frac{\partial L}{\partial \theta_k} = 0 \\ \Rightarrow & \theta_k = \frac{\lambda \sum_{k=1}^{m} \|\mathbf{w}_k\|^2 \sum_{i=1}^{n} \tilde{\beta}_i \langle \Phi_k(\mathbf{x}_i), \mathbf{a}_k \rangle + \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{w}_k, \Phi_k(\mathbf{x}_i) \rangle}{\lambda \sum_{k=1}^{m} \|\mathbf{w}_k\|^2 \sum_{i=1}^{n} \tilde{\beta}_i \|\Phi_k(\mathbf{x}_i)\|^2} \end{aligned} \tag{6.31}$$

By substituting $\mathbf{w}_k = \frac{1}{\lambda}\sum_{i=1}^{n}\theta_k\alpha_i y_i \Phi_k(\mathbf{x}_i)$ and $\mathbf{a}_k = \sum_{i=1}^{n}\tilde{\beta}_i\theta_k\Phi_k(\mathbf{x}_i)$,

$$\lambda\sum_{k=1}^{m}\|\mathbf{w}_k\|^2 = \frac{1}{\lambda}\left(\boldsymbol{\alpha}\circ\mathbf{y}\right)^T K_\theta\left(\boldsymbol{\alpha}\circ\mathbf{y}\right)$$

$$\sum_{i=1}^{n}\alpha_i y_i\langle\mathbf{w}_k,\Phi_k(\mathbf{x}_i)\rangle = \frac{\theta_k}{\lambda}\left(\boldsymbol{\alpha}\circ\mathbf{y}\right)^T K_k\left(\boldsymbol{\alpha}\circ\mathbf{y}\right)$$

$$\sum_{i=1}^{n}\tilde{\beta}_i\langle\Phi_k(\mathbf{x}_i),\mathbf{a}_k\rangle = \theta_k\tilde{\boldsymbol{\beta}}^T K_k\tilde{\boldsymbol{\beta}}$$

where θ_k on the right-hand side is treated as the one in the previous iteration.

We present a scale invariant property of the proposed models which is proved in (Gai et al. [49]).

Lemma 6.7. *(Gai et al., [49]) Multiplying kernel K_θ by a positive constant $\pi > 0$ does not change the optimal value of the objective function.*

We propose Algorithm 6.1 to solve the optimization problem (6.24) by block coordinate descent methods. In step 3 we normalize $\boldsymbol{\theta}$ such that

$$\sum_{k=1}^{m}\theta_k^2 = 1$$

to avoid the explosive increase of $\boldsymbol{\theta}$. As shown in Lemma 6.7, the normalization of $\boldsymbol{\theta}$ does not change the optimization procedure.

Algorithm 6.1: Block coordinate descent methods

Require: feasible $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\theta}$

while optimality conditions are not met **do**

Step 1: Compute $\boldsymbol{\beta}$ according to (6.18)

Step 2: Compute $\boldsymbol{\alpha}$ according to (6.17)

Step 3: Compute $\boldsymbol{\theta}$ according to (6.31) and normalize $\boldsymbol{\theta}$ such that $\sum_{k=1}^{m}\theta_k^2 = 1$

end while

6.3.6 Stopping Criteria

In this section we derive a dual feasible objective value of the problem (6.24) for a stopping criteria of the algorithm. For this purpose, we use the dual form of W-MWD in (6.14), since the SVM part and the SVDD part are independent each other and can be treated separately. At first we derive the dual form for a fixed $\boldsymbol{\theta}$. In the notation in

(6.14) we put the constraints of the SVM part into S and the constraint of the SVDD part into $M \setminus S$.

$$S : \begin{cases} y_i(\sum_{k=1}^{m} \langle \mathbf{w}_k, \theta_k \Phi_k(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, & i = 1, \ldots, n \\ \xi_i \geq 0, \quad i = 1, \ldots, n \end{cases}$$

$$M \setminus S : \sum_{k=1}^{m} \theta_k^2 \left\| \Phi_k(\mathbf{x}_i) \right\|^2 - 2 \sum_{k=1}^{m} \langle \theta_k \Phi_k(\mathbf{x}_i), \mathbf{a}_k \rangle + \sum_{k=1}^{m} \left\| \mathbf{a}_k \right\|^2 \leq R, \; i = 1, \ldots, n$$

For a fixed $\boldsymbol{\theta}$, W-MWD in (6.14) is written as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda R} \left(\boldsymbol{\alpha} \circ \mathbf{y} \right)^T K_\theta \left(\boldsymbol{\alpha} \circ \mathbf{y} \right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq 1, \quad i = 1, \ldots, n \\ & \sum_{i=1}^{n} \tilde{\beta}_i K_{\theta,ii} - \tilde{\boldsymbol{\beta}}^T K_\theta \tilde{\boldsymbol{\beta}} - R \geq 0. \\ & \sum_{i=1}^{n} \tilde{\beta}_i = 1, \quad 0 \leq \tilde{\beta}_i, \quad i = 1, \ldots, n. \end{aligned} \tag{6.32}$$

Now we consider the dual problem as a function of $\boldsymbol{\theta}$:

$$F(\boldsymbol{\theta}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda R(\boldsymbol{\theta})} \left(\boldsymbol{\alpha} \circ \mathbf{y} \right)^T K_\theta \left(\boldsymbol{\alpha} \circ \mathbf{y} \right) \tag{6.33}$$

$$G(\boldsymbol{\theta}) = \sum_{i=1}^{n} \tilde{\beta}_i K_{\theta,ii} - \tilde{\boldsymbol{\beta}}^T K_\theta \tilde{\boldsymbol{\beta}} - R(\boldsymbol{\theta}) \tag{6.34}$$

where

$$R(\boldsymbol{\theta}) = \text{mean}_{\tilde{\beta}_i > 0} \left(K_{\theta,ii} - 2\tilde{\boldsymbol{\beta}}^T K_{\theta,i\cdot} + \tilde{\boldsymbol{\beta}}^T K_\theta \tilde{\boldsymbol{\beta}} \right).$$

Since we are assuming $R > 0$, $\boldsymbol{\theta} \neq 0$ (at least one element of $\boldsymbol{\theta}$ is not zero). From Lemma 6.7, scaling of kernels ($\theta_k^2, \; k = 1, \ldots, m$) does not affect the solution. Therefore we add a constraint of $\sum_{k=1}^{m} \theta_k^2 = 1$ and solve the following optimization problem for a fixed $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda R(\boldsymbol{\theta})} \left(\boldsymbol{\alpha} \circ \mathbf{y} \right)^T K_\theta \left(\boldsymbol{\alpha} \circ \mathbf{y} \right) \\ \text{s.t.} \quad & \sum_{k=1}^{m} \theta_k^2 = 1 \\ & \sum_{i=1}^{n} \tilde{\beta}_i K_{\theta,ii} - \tilde{\boldsymbol{\beta}}^T K_\theta \tilde{\boldsymbol{\beta}} - R(\boldsymbol{\theta}) \leq 0 \end{aligned} \tag{6.35}$$

where

$$R(\boldsymbol{\theta}) = \text{mean}_{\tilde{\beta}_i > 0} \left(K_{\theta,ii} - 2\tilde{\boldsymbol{\beta}}^T K_{\theta,i\cdot} + \tilde{\boldsymbol{\beta}}^T K_\theta \tilde{\boldsymbol{\beta}} \right).$$

Note that in the last inequality $G(\boldsymbol{\theta}) \leq 0$ (not $G(\boldsymbol{\theta}) \geq 0$), because $\tilde{\beta}_i \geq 0$ and $g_i(\mathbf{x}) \leq 0$, $i = 1, \ldots, m$ in the optimization problem P in (2.13).

Let $u_k = \theta_k^2$ for $k = 1, \ldots, m$. $\frac{1}{2\lambda R} (\boldsymbol{\alpha} \circ \mathbf{y})^T K_\theta (\boldsymbol{\alpha} \circ \mathbf{y})$ is of the form of a homogeneous linear fractional problem $\frac{\mathbf{a}^T \mathbf{u}}{\mathbf{b}^T \mathbf{u}}$, which shows the scaling of $\mathbf{u}$ does not affect the optimal value of the solution. Substituting $\sum_{k=1}^m u_k = 1$, it becomes of the form of a linear fractional programming problem $\frac{\mathbf{c}^T \mathbf{u} + \alpha}{\mathbf{d}^T \mathbf{u} + \beta}$. Since it is both pseudoconvex and pseudoconcave (Cambini and Martein, [22]), a local minimum is the global optimal solution. Linear fractional programming problems are transformed to linear programming problems and can be solved with standard linear problem solvers (Saha et al, [132]).

By substituting $\boldsymbol{\theta}$ into (6.35), we obtain a dual objective value of (6.24). From the weak duality it guarantees that the optimal solutions in (6.24) are bounded from below. We can use the dual objective values to compute the duality gap as a stopping criteria.

6.4 Experiments

In this section we conduct experiments to check the performance of the proposed method. The purpose of the experiments is to compare the prediction accuracy of the proposed method with ℓ_p-norm MKL.

6.4.1 Experiments with Benchmark Datasets

In this section we conduct experiments for binary classification using benchmark datasets.

Experimental Design

The main purpose of the experiments is to compare the performance of the proposed method with ℓ_p-norm MKL. We also compare the performance of the proposed method with the MKL method in (Gai et al. [49]), which we denote G-MKL.

1. **WR-MKL** proposed MKL:

2. ℓ_p**-norm MKL** in (6.3) in Section 6.2.1:
   We show the results for $p \in \{1, 2, 4, 8\}$.

3. **SVM** SVM with fine tuned parameters
   The settings for hyper-parameters and α is the same as the SEB method with Gaussian kernels in Chapter 3.

4. **G-MKL** MKL method in (Gai et al. [49]):
   G-MKL also uses the block coordinate method to derive the optimal solutions. The first step and second step is the same as Algorithm 6.1. In the third step, for the computation of $\boldsymbol{\theta}$ the authors propose the gradient descent method for the following function:

$$\min_{\boldsymbol{\theta}:} \sum_i \alpha_i - \frac{1}{2r(\boldsymbol{\theta})} \sum_{i,j} \alpha_i \alpha_j y_i y_j K_\theta(i,j) \tag{6.36}$$

$$\text{where } r(\boldsymbol{\theta}) = \sum_i \beta_i K_\theta(i,i) - \sum_{i,j} \beta_i \beta_j K_\theta(i,j) \tag{6.37}$$

   K_θ is the kernel matrix $K_\theta = \sum_{k=1}^m \theta_k K_k$.

We prepare a set of kernels for all features and for each single feature. For each of those kernels we prepare a set of Gaussian kernels with different width parameters $\sigma^2 \in \{2^{-8}, 2^{-6}, \ldots, 2^{10}\}$. In total, we use a linear combination of $d+1$ (features) $\times$ 10 (width parameters) $= 10(d+1)$ kernels where d is the number of features.

In the first sampling we set $\log \lambda \in \{-10, -8, \ldots, 8\}$ and the α-percent region to be 20, which is equivalent to the parameter setting for C and α-percent region in the experiments with Gaussian kernel in Chapter 3. We set the size of the datasets as the "(Training/Validation/Test) with Training $= 100$" column in Table 1.2. Experiments are repeated 50 times for each dataset.

Tables 6.1 report the results of binary classification carried out on the 17 benchmark datasets. In the tables, the first row for each dataset shows the prediction accuracy and the best results are emphasized in boldface. The second row shows the percent of nonzero coefficient $\theta_k > 1e-5$ in the optimal combination of kernels. We conduct the two-tailed paired t-test for each run of experiments to compare the prediction accuracy of WR-MKL to that of the best of ℓ_p-norm MKL for $p \in \{1, 2, 4, 8\}$. If the two methods are significantly different (p-values < 0.05), (*) is shown beside the number in the column of the better method. We also conduct the two-tailed paired t-test for each run of experiments to compare the prediction accuracy of WR-MKL to that of G-MKL. If the two methods are significantly different (p-values < 0.05), ($^{(*)}$) is shown beside the number in the column of the better method.

Table 6.1: Prediction accuracy for binary classification experiments

Dataset	**SVM**	**G-MKL**	ℓ_1**-norm**	ℓ_2**-norm**	ℓ_4**-norm**	ℓ_8**-norm**	**WR-MKL**
$^{(*)}$ Banknote	99.54 ± 0.54	98.46 ± 1.02 (54%)	98.69 ± 1.10 (32%)	98.70 ± 0.93 (100%)	98.65 ± 1.00 (100%)	98.62 ± 1.05 (100%)	**98.74** ± 1.17 (35%)
Credit	80.84 ± 1.18	80.59 ± 4.29 (41%)	**81.50** ± 1.65 (11%)	80.26 ± 8.26 (99%)	78.78 ± 11.62 (100%)	79.62 ± 8.48 (100%)	81.30 ± 1.93 (33%)
Crowd-sourced	91.37 ± 1.28	89.86 ± 1.55 (55%)	89.53 ± 1.23 (17%)	**90.82*** ± 0.97 (99%)	90.76 ± 1.09 (100%)	90.81 ± 1.10 (100%)	89.88 ± 1.30 (24%)
Drive	75.16 ± 2.79	78.34 ± 13.81 (29%)	80.88 ± 2.72 (12%)	73.32 ± 6.42 (98%)	68.66 ± 4.85 (100%)	65.33 ± 5.70 (100%)	**81.65**$^{*,(*)}$ ± 2.13 (31%)
German	71.81 ± 2.13	70.98 ± 1.89 (57%)	**71.72** ± 2.08 (28%)	71.60 ± 2.12 (97%)	71.39 ± 2.12 (98%)	71.68 ± 2.19 (98%)	71.49 ± 1.98 (52%)
Letter	72.23 ± 2.14	70.28 ± 2.14 (52%)	**72.34*** ± 2.06 (21%)	72.11 ± 1.75 (100%)	72.21 ± 1.89 (100%)	72.20 ± 1.91 (100%)	$72.04^{(*)} \pm 1.96$ (34%)
MAGIC	81.24 ± 1.49	78.82 ± 1.87 (38%)	78.82 ± 1.87 (21%)	79.61 ± 1.61 (99%)	**79.78*** ± 1.82 (100%)	79.58 ± 1.80 (100%)	78.25 ± 1.92 (34%)
Occupancy	98.44 ± 0.53	98.67 ± 0.33 (30%)	98.47 ± 0.58 (20%)	98.54 ± 0.49 (100%)	**98.70** ± 0.36 (100%)	98.68 ± 0.33 (100%)	98.68 ± 0.40 (37%)
Opt Digit	95.44 ± 1.47	86.57 ± 15.14 (64%)	87.00 ± 3.74 (12%)	86.21 ± 7.59 (96%)	82.94 ± 13.03 (97%)	82.30 ± 13.32 (97%)	**89.42**$^{*,(*)}$ ± 2.02 (15%)
Page Blocks	93.98 ± 0.63	93.22 ± 1.33 (50%)	**92.97** ± 3.40 (18%)	92.93 ± 1.60 (99%)	92.53 ± 1.68 (100%)	92.54 ± 1.53 (100%)	92.80 ± 1.13 (33%)
Pen Digit	95.43 ± 1.20	89.04 ± 2.31 (48%)	93.94 ± 1.46 (15%)	89.19 ± 1.99 (100%)	88.81 ± 1.82 (100%)	88.77 ± 1.82 (100%)	**94.02**$^{(*)}$ ± 1.40 (24%)
Satellite	93.21 ± 0.77	91.48 ± 1.09 (49%)	91.24 ± 1.31 (12%)	92.33 ± 0.96 (100%)	92.45 ± 0.89 (100%)	**92.46*** ± 0.88 (100%)	91.40 ± 1.38 (19%)
Segment	91.62 ± 1.45	93.72 ± 1.42 (60%)	93.45 ± 2.23 (16%)	92.05 ± 5.29 (98%)	90.92 ± 5.74 (99%)	90.07 ± 6.36 (99%)	**94.06*** ± 1.60 (25%)
Splice	79.44 ± 1.69	88.57 ± 1.42 (57%)	87.52 ± 1.53 (15%)	**87.93** ± 1.62 (99%)	86.57 ± 1.77 (100%)	85.70 ± 1.83 (100%)	87.92 ± 1.80 (11%)
Statlog	98.48 ± 0.89	95.93 ± 7.59 (52%)	97.53 ± 4.14 (29%)	97.64 ± 3.92 (98%)	96.68 ± 3.79 (100%)	95.56 ± 4.34 (100%)	**97.88** ± 2.67 (62%)
Svmguide1	95.55 ± 0.54	95.78 ± 0.85 (47%)	95.51 ± 0.91 (40%)	95.86 ± 0.74 (100%)	**95.93** ± 0.62 (100%)	95.89 ± 0.55 (100%)	95.81 ± 0.71 (25%)
Wine Quality	79.57 ± 0.88	78.87 ± 0.97 (65%)	79.22 ± 0.71 (17%)	79.26 ± 0.86 (100%)	79.27 ± 0.86 (100%)	**79.35** ± 0.89 (100%)	79.14 ± 0.67 (41%)

6.4.2 Experiments with Synthesized Data

To examine the properties of the proposed method we conduct experiments using synthesized data with various levels of sparsity. The datasets are the same as those in (Kloft et al., [77]) which are synthesized as follows. The input data $\mathbf{x}$ have 50 features ($\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,50})$). Each feature $x_{i,k}$ is generated from the independent Gaussian distribution with the equal variance 1 and the mean $1.75\rho_k$ for the label 1 and $-1.75\rho_k$ for the label -1 where the ρ_k takes the value either of 0 or 1.

In the most sparse data only one ρ_k is set to 1 and the rest are set to 0; in other words, only one feature has discrimination power and the rest are noise. In the least sparse data set, all 50 ρ_ks are set to 1; that is, all the features are relevant. When more than one feature are relevant, all of them are required to achieve the maximum prediction accuracy. There are a total of six data sets and the ratio of noisy features (to the total number of features which is 50) for the six data sets are $\{0, 0.44, 0.66, 0.82, 0.92, 0.98\}$.

We train the model using the training datasets of 50 instances. We construct a set of kernels for all features and for each single feature. For each of those kernels we prepare a set of Gaussian kernels with different width parameters $\sigma^2 \in \{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$. In total, we use a linear combination of 51 (features) $\times$ 21 (width parameters) $= 1071$ kernels. The regularization parameter $\lambda \in \{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$ is tuned by the validation datasets of $5,000$ instances. We simply choose the value of λ with the highest accuracy on the validation data. Finally we evaluate the prediction accuracy of the proposed method by the test datasets of $5,000$ instances and compare the results with those of the ℓ_p-norm methods with $p \in \{1, 2, 4, 8, 16\}$. Those numbers are chosen this way so that the results are comparable with the original paper. We repeat the experiments 100 times for each dataset.

In the experiments we use normalized kernels which are normalized by the multiplicative normalization method (Kloft et al., [77]):

$$k(x, x') \Rightarrow \frac{k(x, x')}{\frac{1}{n}\sum_{i=1}^{n} k(x_i, x_i) - \frac{1}{n^2}\sum_{i,j=1}^{n} k(x_i, x_j)}. \tag{6.38}$$

6.4.3 Results and Discussion

Both experiments with benchmark datasets and synthesized datasets show that the performance of the proposed method in terms of classification accuracy is comparable to the best of ℓ_p-norm MKL with the fine tuned parameter p. In Table 6.1, the results of WR-MKL are significantly better than the best of ℓ_p-norm MKL for three datasets and significantly worse for four datasets out of 17 datasets. Fig. 6.1 is the prediction accuracy of the six methods on the synthesized datasets. The proposed method shows the comparable prediction accuracy to the best of ℓ_p-norm methods for $p \in \{1, 2, 4, 8, 16\}$ over all types of data.

Both WR-MKL and ℓ_1-norm methods return sparse coefficients of kernels and the results of WR-MKL are similar to those of ℓ_1-norm methods. We recorded the average iteration number for the convergence. The mean iteration number of the block coordinate descent for all 17 datasets is 797.1 for ℓ_1-norm, 162.9 for ℓ_2-norm, 251.9 for

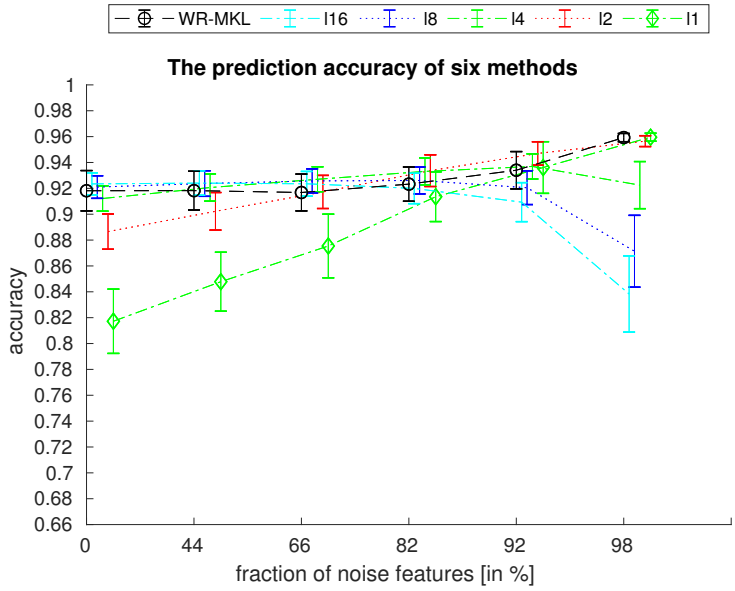


Figure 6.1: The prediction accuracy of the six methods on synthesized datasets

ℓ_4-norm, 287.9 for ℓ_8-norm and 659.8 for WR-MKL. Therefore ℓ_1-norm and WR-MKL is comparable in this respect too. Experiments with synthesized data indicate that WR-MKL is more robust than l1-norm method for the data with various levels of noisy features.

We also compared the results of WR-MKL with those of G-MKL. In Table 6.1, the results of WR-MKL are significantly better than G-MKL for four datasets out of 17 datasets. The objective function of $\boldsymbol{\theta}$ in (6.36) has the form of linear fractional programming problem $\frac{\mathbf{c}^T\boldsymbol{\theta}+a}{\mathbf{d}^T\boldsymbol{\theta}+b}$. Since it does not have the Lipschitz property and it is not bounded, the convergence to the local optimal solutions is not guaranteed (Bertsekas [11]). Indeed applying the gradient descent method in (6.36), the duality gap computed by the method in Section 6.3.6 did not approach zero.

As a reference we show the results of SVM. In general the MKL methods do not necessary improve the accuracy of SVM with fine-tuned parameters. As suggested in (Kakade et al. [68]), the generalization of error rates of MKL may increase logarithmically as the number of kernels.

As shown in Section 6.2.1, in order to construct a MKL model we replace the inner product $\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle$ by a sum $\sum_{k=1}^m \langle \mathbf{w}_k, \sqrt{\theta_k}\Phi_k(\mathbf{x}_i)\rangle$. However, to avoid the non-convexity arising from $\sqrt{\theta_k}$, $\sqrt{\theta_k}\mathbf{w}_k$ is substituted by $\hat{\mathbf{w}}_k$. (If we use θ_k instead of $\sqrt{\theta_k}$, $\theta_k\mathbf{w}_k$ is still non-convex for $(\theta_k, \mathbf{w}_k)$.) Then, the MKL optimization problem is formulated as (6.2). ℓ_p-norm MKL solves the optimization problem using the additional

constraint:

$$\|\boldsymbol{\theta}\|_p^2 \leq 1, \quad p \geq 1 \ .$$

The cost for the substitution $\sqrt{\theta_k}\mathbf{w}_k$ by $\hat{\mathbf{w}}_k$ is that it loses the Lipschitz property. The equivalent primal formulation (6.4) can not be solved directly by the SGD method (Section 2.3.5) and it is solved using a linear approximation in online learning.

In our approach we incorporate a radius into the MKL formulation (6.25):

$$\min_{\boldsymbol{\theta},\mathbf{w},b,\mathbf{a}} \frac{\lambda}{2} \sum_{k=1}^{m} \|\mathbf{w}_k\|^2 \max_i \sum_{k=1}^{m} \|\theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k\|^2 + \sum_{i=1}^{n} \max\left(0, 1 - y_i(\sum_{k=1}^{m} \langle \mathbf{w}_k, \theta_k \Phi_k(\mathbf{x}_i)\rangle + b)\right)$$

The radius is a quadratic function of $\boldsymbol{\theta}$ and $\mathbf{a}$. It is not a convex function, but it is still strongly convex with respect to each $\boldsymbol{\theta}$, $\mathbf{w}$ and $\mathbf{a}$ and keeps the Lipschitz property.

We think that our approach will bring important advantages in the light of Theorem 2.2 and Theorem 2.7. To clarify the point we would like to discuss a slightly different model which we are currently working on in Appendix A.1 and in Section 8.2.3.

6.5 Summary

In this chapter we formulated the combined model of SVM and SVDD as a standard one-level optimization problem. We prove the existence of global optimal solutions and derive the dual feasible objective values which are used as a stopping criteria.

Experiments showed that the performance of the proposed method is comparable to the ℓ_p-norm method with fine tuned p. The proposed method produces sparse solutions. Although the convergence speed is slower than ℓ_p-norm methods, it is more robust than l1-norm method for various types of data and it does not need to adjust the additional parameters.

Contributions and achievements:

- We formulated a new WR-MKL model as a standard one-level optimization problem and proved the existence of a unique global solution.
- We proposed a numerical algorithm using the block coordinate descent method and derived the dual feasible objective values as the stopping criteria.
- The proposed MKL model dose not require any additional parameters.

- We experimentally showed the proposed method has good prediction accuracy and robustness compared to the existing algorithms.

Chapter 7

Optimal Kernel Construction

Kernel functions form the geometry of the feature space. Designing of kernel functions manually is an expensive task and requires domain-specific knowledge. In this chapter, we propose a new method to automatically construct kernel functions. We achieve this by searching for optimal subsets of features and optimal combination of primitive kernels.

7.1 Introduction

Since the invention of MKL by Lanckriet et al. [85], the study of MKL has mainly focused on the linear combination of kernels except for a few articles (Cortes et al. [28], Varma and Babu [162], Meirom and Kisilev [105]). A common practice in MKL is to prepare a set of kernels which consists of kernels corresponding to each individual feature and a kernel corresponding to all the features. Therefore, the vast space of nonlinear combinations of kernels which correspond to arbitrary subsets of features still remains unknown.

In order to search for the best subsets of features, we consider a heuristic method to explore this vast space. We apply the methods developed in the previous chapters. We mainly examine the two methods—a heuristic method based on MKL in Chapter 6 and a binary PSO developed in Chapter 4.

In order to find an optimal kernel, we construct one using Genetic Programming (GP). The space of kernel matrices are closed under the operations of addition, pointwise product and exponentiation (Section 2.3.3). GP has a tree-based representation. Operations defined on the functional nodes (internal nodes) make it possible to explore the space by constructing new kernels using these three arithmetic operations.

We make use of the versatility and the flexibility of GP to explore the space of linear and nonlinear combinations of kernels.

Gap or weakness in previous research: There have been many attempts at constructing optimal kernels through linear and non-linear combination of basic kernels. An optimal kernel requires an optimal subset of features, however none of the existing methods examine combination of kernels with different feature subsets.

Objectives of Chapter 7: The goal is to incorporate the feature selection process into the kernel construction process.

7.2 Related Work

In this section, we review the relevant concepts and previous works in the literature.

7.2.1 Optimal Kernel Construction

GP has been used to automatically discover a new form of kernel functions using reproduction, crossover and mutations. The evolved kernels are expressed as tree structures in which internal nodes correspond to functional operations such as $+$ and $\times$ and leaf nodes correspond to input vectors (Howley and Madden [66]; Dioşan et al. [33]; Koch et al. [78]) or kernels (Sullivan and Luke [151]; Dioşan et al. [34]), so that the obtained expressions satisfy Mercer's condition in Theorem 2.4.

Koch et al. [78] search for the optimal hyper-parameters using their software which implements genetic methods such as SPOT (Bartz-Beielstein et al. [6]) and TDM (Konen et al. [80]), and construct kernels using operators $(+, \times, \exp(\cdot))$ and input vectors. They conclude their experiments with the remark; "Our method rediscovered multiple standard kernels, but no significant improvements over standard kernels were obtained." Sullivan and Luke [151] use standard kernels (Polynomial, Gaussian, Sigmoid) instead of input vectors to construct composite kernels. In the tree representation of GP, the leaf nodes are input vectors, the internal nodes that are one level above the leaf nodes are the kernel functions, and the internal nodes that are two or more levels above the leaf nodes are functions $(+, \times, \exp(\cdot))$. Dioşan et al. [34] adopt a simpler configuration in which leaf nodes are standard kernels and random constants, and internal nodes are functions $(+, \times, \exp(\cdot))$.

7.3 Proposed System

We propose a system to automatically construct optimal kernels, which consists of two parts:

1. a subsystem to find the optimal subsets of features (using an embedded or wrapper); and
2. a subsystem that takes the optimal subsets of features and then finds an optimal combination of kernels (using GP).

A feature of our proposal is the combination of feature selection subsystem and the optimal kernel construction subsystem (Figure 7.1). We also use the weights of kernels of MKL as the prior distribution of leaf nodes in initial population in GP. We send the kernels along with their weights to the GP-based kernel construction subsystem. In GP the leaf nodes are selected in proportion to the relative weights of the kernels.

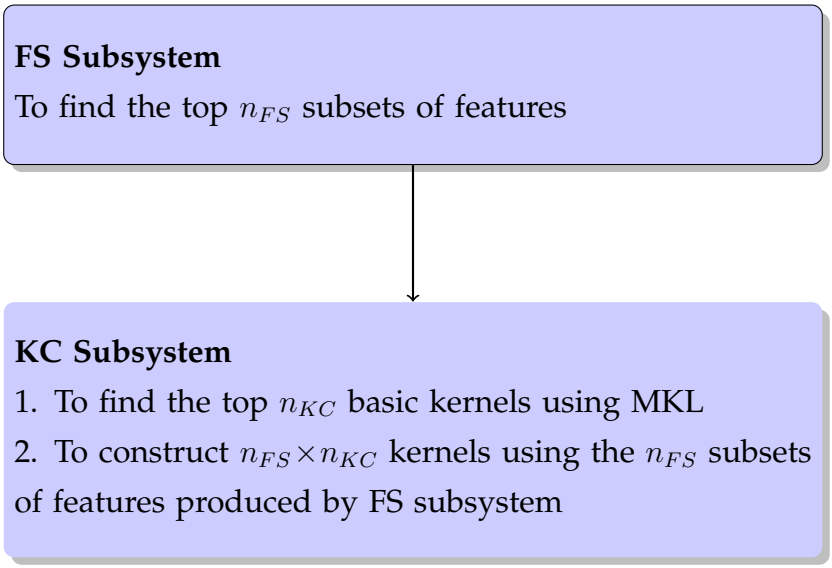


Figure 7.1: Main Operations in Subsystems

7.3.1 Feature Selection (FS) Subsystem

Let's start with the definition of Group Feature Selection. Features are elements in the input vector $\mathbf{x} = (x_1, \ldots, x_d)$. Feature selection is an operation to select a subset F_k of features from $\{x_1, \ldots, x_d\}$. For instance, $F_1 = \{x_1, x_4, x_5\}$, $F_2 = \{x_2, x_3\}$ and

$F_3 = \{x_3, x_5, x_7\}$. Group feature selection is an operation to select a union of subsets $\bigcup_{k\in\mathcal{K}} F_k$ (Figure 7.2). For instance $\bigcup_{k=1}^{3} F_k = \{F_1, F_2, F_3\}$. That is, group feature selection evaluates a union of feature subsets.

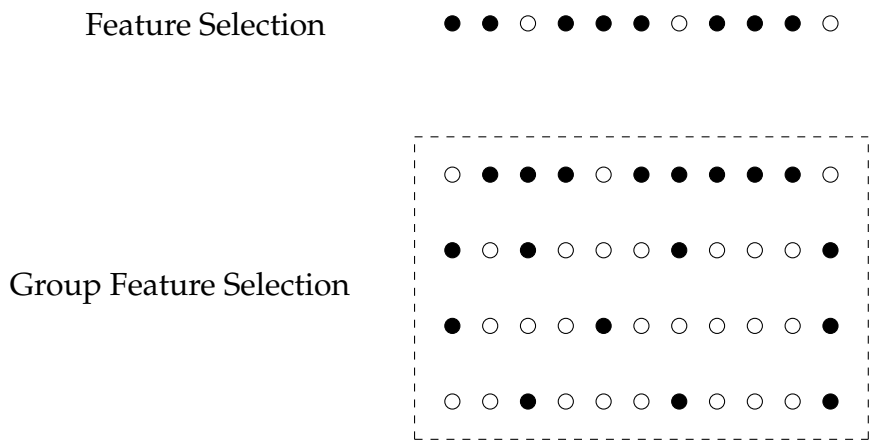


Figure 7.2: Group Feature Selection

Feature Selection (FS) subsystem takes input vectors with d features and outputs the top n_{FS} subsets of features out of possible $2^d - 1$ subsets of features, which will be used in the kernel construction subsystem. Basically any feature selection methods in Section 2.10.2 can be used for this purpose.

7.3.2 Kernel Construction (KC) Subsystem

This subsystem consists of two parts. In the first part, we construct basic kernels. Each basic kernel has a set of hyper-parameters that must be specified. A large number of basic kernel functions with various hyper-parameter settings are constructed and used in an MKL model to determine their relative importance (based on the magnitude of the corresponding coefficients). The n_{KC} top basic kernels are chosen. Then, we construct $n_{FS} \times n_{KC}$ kernels combining the n_{FS} subsets of features produced by the feature selection subsystem. Lastly, we use the MKL method again with $n_{FS} \times n_{KC}$ kernels to compute the relative weights of kernels, which are later used as the probabilities of selecting variable terminals in GP.

One reason for attempting to make the distribution non-uniform is that kernels with a larger number of parameters generate a larger number of basic kernels, and would therefore be more likely to be selected if the terminal nodes were chosen uniformly.

In the second part, we use GP to evolve individuals that are composite kernel functions and evaluate to kernel matrices. The GP system consists of the following components.

i) Primitive functions:

There are three functions which maintain the positive definiteness of the kernels (Section 2.3.3 and implement the following mappings:

- the $+$ function returns the sum of the kernels: $K_1 + K_2$;
- the $\times$ function returns the Hadamard (pair-wise) product of the kernels: $K_1 \circ K_2$; and
- the $\exp$ function returns the exponential of the kernel: $\exp(K)$.

ii) Variable terminals:

All the terminals are basic kernel functions of one of the two types of kernels, Gaussian kernel (2.17) and Polynomial kernel (2.18) in Section 2.3.3.

iii) Constant terminals:

Random constants, i.e. uniformly generated random numbers in $[0, 1]$ (Koza [81])

iv) Fitness function:

As before, this function maps an individual to its fitness which is in the range $[0, 1]$. An individual is a kernel function. An SVM model is constructed using this kernel function. The model is trained on the training data by solving the optimization problem (2.8). The prediction accuracy of the model on the validation set is returned as the fitness of the individual.

After the last generation the best individual (i.e. a kernel function) is returned.

7.3.3 Procedures of Kernel Construction

In the experiments we compare three types of kernel construction based on; only FS-subsystem, only KC-subsystem and both FS-subsystem and KC-subsystem. We explain in detail how to implement those methods (Figure 7.3).

i) FS subsystem:

We construct the kernel with the best feature subset in the following way:

1. Firstly we run the SVM to determine the optimal hyper parameters.
2. Then we determine the top n_{FS} feature subsets.
3. Lastly we evaluate the kernel with the best feature subset using test data.

ii) KC subsystem:

We construct the optimal kernel with all features in the following way:

1. We construct a set of basic kernels with various kernel hyper-parameters using a given set of kernels.
2. Then we run MKL with those kernels and determine the top n_{KC} kernels by comparing the coefficients of kernels. This process also determines the optimal specification of regularization hyper-parameter.
3. Lastly we run the GP to construct optimal kernels from these basic kernels. We also send to GP the information of the weight of kernels and the optimal specification of the regularization hyper-parameter.

iii) FS-KC subsystem:

Using the information from those two subsystems we construct the optimal kernel with the best subsets of features in the following way:

1. Using the n_{FS} feature subsets from FS subsystem and the n_{KC} kernels from KC subsystem, we construct $n_{FS} \times n_{KC}$ basic kernels.
2. Then we run MKL to determine the weight of kernels. This process also determines the optimal specification of the regularization hyper-parameter.
3. Lastly we run the GP to construct optimal kernels from these basic kernels. We also send to GP the information of the weight of kernels and the optimal specification of the regularization hyper-parameter.

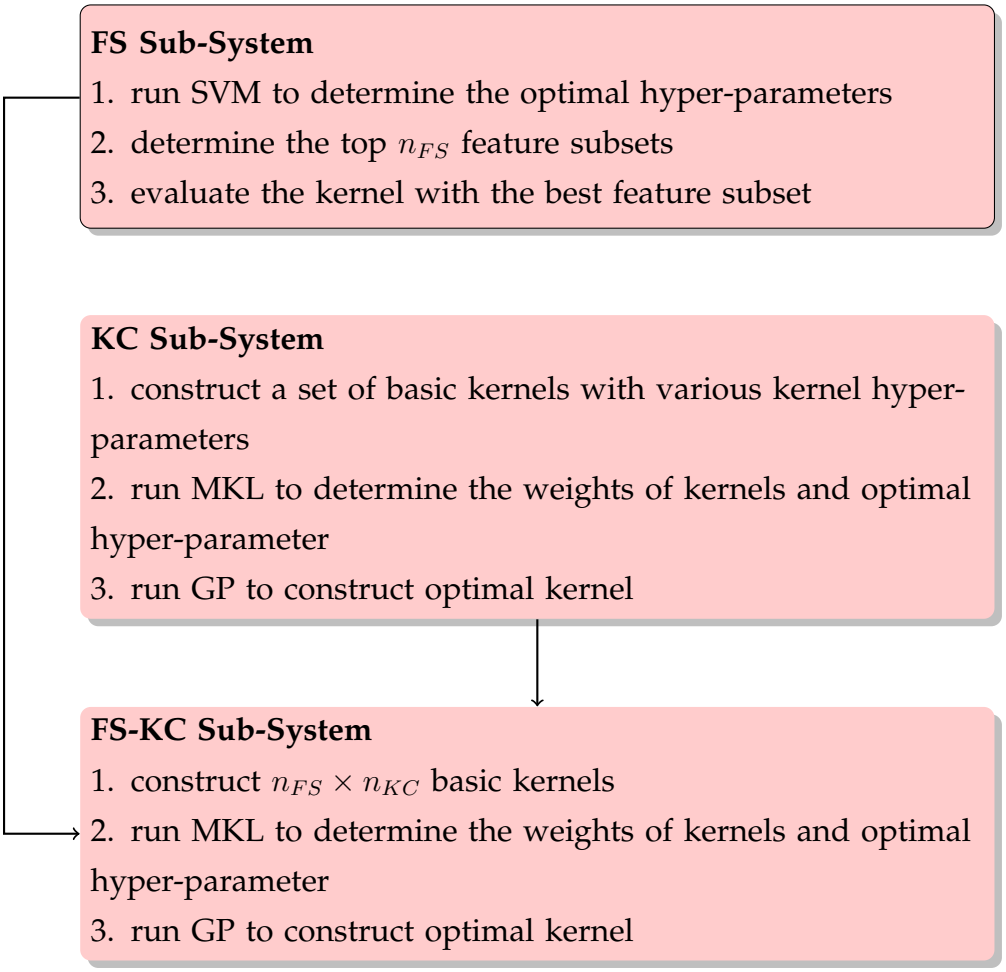


Figure 7.3: Kernel Construction Process

7.4 Experiments

In this section, we conduct experiments for group feature selection and for optimal kernel construction. The experiments for group feature selection are the preliminary ones and determine the group feature selection method which will be used in the experiments of optimal kernel construction.

7.4.1 Experiments for Group Feature Selection

In this section we start the group feature selection experiments with a wider scope. Firstly we conduct experiments to compare the binary PSO developed in Chapter 4 with GA (Section 2.6), which is also suitable for the task of group feature selection. Next we repeat the experiment with K-Nearest Neighbor (KNN) method as the classifier instead of SVM. We also conduct experiments to compare a traditional feature selection method (SFS and SBS) in Section 2.10.2 with the best EC-based group feature selection method. Lastly we conduct experiments to compare a heuristic method based on WR-MKL (Chapter 6) with the best EC-based group feature selection method. The former method represents the embedded approach in which we use the coefficients of kernels to evaluate the relative importance of features. The latter method represents the wrapper approach in which we use the EC method to explore the space of feature subsets and directly evaluate each subset. After examining these two approaches we will choose better one for the experiments of optimal kernel construction.

Heuristic method based on WR-MKL (H-MKL):

We propose a heuristic method based on MKL method, which uses the kernel coefficients to evaluate the relative importance of variables (features). For an input vector $\mathbf{x}$ with d features, let $F = \{1, \ldots, d\}$ be the set of all features. It requires to specify the number N_F of subsets of features returned by the algorithm. It also requires to specify the number of features n_{pw} whose power sets are examined by the method.

Firstly, we prepare a set of kernels each of which corresponds to each single feature. We run MKL to evaluate the relative importance of the kernels and select top n_{pw} features $F_1 \subset F$. For those important features we evaluate all $2^{n_{pw}} - 1$ subsets (the power set) of features. We run MKL again to select the top N_F subsets of features out of $2^{n_{pw}} - 1$ subsets. Let $F_2 = F \setminus F_1$ be the rest of $(d - n_{pw})$ features. We pick up a feature $f \in F_2$ and construct $2N_F$ subsets of features, which consist of a copy of N_F subsets of features and a copy of N_F subsets of features with the feature f added to

each subset. We select the top N_F subsets of features out of $2N_F$ subsets. We repeat the process until we exhaust all features in F_2. The total number of the MKL run is $d - n_{pw} + 1$. The details of this procedure can be found in Algorithm 7.1.

Algorithm 7.1: Heuristic method based on WR-MKL

Require: N_F the number of best features returned; n_{pw} the number of best features whose power set is examined

Preparation:

1. determine the optimal specification of hyper-parameters;
2. prepare a set of kernels each of which correspond to each feature;
3. run WR-MKL to determine the relative importance of features
4. select top n_{pw} features F_1 and construct $2^{n_{pw}} - 1$ kernels corresponding to the power set of n_{pw} features
5. run WR-MKL and select top N_F subsets of features
6. let $F_2 = F \setminus F_1$ be the rest of $d - n_{pw}$ features

for $f \in F_2$ **do**

1. construct $2N_F$ kernels which consists of a copy of N_F subsets of features and a copy of N_F subsets of features with the feature f added to each subset
2. run WR-MKL and select top N_F subsets of features

end for

We denote this method H-MKL in the following experiments.

Experiments of Group Feature Selection (GA vs. Binary PSO)

In the experiment, we use Gaussian kernel in (2.17) in Section 2.3.3. The range and the step size of hyper-parameters and the value of α are the same as the experiments in Chapter 3.

We compare the prediction accuracy of GA, B-SPSO and B-FLRPSO.

1. **GA** (Burjorjee [21]):
   We implement GA software "SpeedyGA" with the following default settings:
   - The probability of cross over $= 1$
   - The mutation probability (per bit) $= 0.003$
   - Sigma Scaling

The fitness value $f(i,t)$ for an individual i is transformed to:

$$f_{new}(i,t) = \begin{cases} \frac{f(i,t)-\mu_t}{\sigma_t} & \text{if } \sigma_t \neq 0 \\ 1 & \text{if } \sigma_t = 0 \end{cases}$$

where μ_t is a mean of $f(\cdot,t)$ and σ_t is a standard deviation of $f(\cdot,t)$. The purpose of sigma scaling is to prevent a premature convergence (Mitchell [109]).

- Stochastic Universal Sampling (SUS)
  The roulette wheel sampling often results in a large deviation between the actual allocation and the expected allocation. SUS is proposed to minimize the deviation. Suppose that the population size $= n$. In order to select N parents, SUS divides the $(0,1)$ interval into n parts in proportion to the fitness value of each individual, put N equally spaced pointers in $(0,1)$ and select an individual k times where k is the number of the pointers fall in the portion of the individual (Mitchell [109]).

2. **B-SPSO** in Section 4.3.3:
   We set $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$. We also set the parameter ξ so that ξ is 0.08 at the first iteration and lineally decreases to 0.01 at the last iteration which is the same setting as in the experiments for binary PSO in Chapter 4.

3. **B-FLRPSO** in Section 4.3.4:
   We set $(\omega, c_1, c_2, c_3) = (0.5, 1.3, 1.3, 1.3)$. We set the ratio of preservation (RP) so that RP starts with 0.95 at the first iteration and lineally decreases to 0.99 at the last iteration which is the same setting as in the experiments for binary PSO in Chapter 4.

At first we run SVM with all features and determine the optimal specifications of σ^2 and C. Using the optimal hyper-parameters, we examine three patterns of population size and the number of generations for each method as follows.

- population size = 6 and number of generations = 5
  (Since the GA program (SpeedyGA) assumes that the population size is even, we set it to 6.)

- population size = 10 and number of generations = 10

- population size = 20 and number of generations = 20

Then we record top 200 feature subsets and report the following results using test data.

- Best: evaluation for the best feature subset estimation
- Mean25: mean of the test evaluations for the top 25 group feature selection
- Mean50: mean of the test evaluations for the top 50 group feature selection
- Mean100: mean of the test evaluations for the top 100 group feature selection
- Mean200: mean of the test evaluations for the top 200 group feature selection

Table 7.2 lists the benchmark datasets used in this experiments. We have chosen 10 datasets with larger features from Table 1.2. Experiments are repeated 50 times for each dataset.

Table 7.1: Datasets

Dataset	Instances	Features	(training/validation/test)	Classes
Default of credit card clients	30000	24	(100/1000/3000)	
Crowd-sourced Mapping Data Set	10845	28	(100/1000/3000)	$= 2$ vs. $\neq 2$
Sensorless Drive Diagnosis	58509	49	(100/1000/3000)	$<= 5$ vs > 5
German	1000	24	(100/400/500)	
Letter	15000	16	(100/1000/3000)	$<= 13$ vs > 13
Optical Handwritten Digits	5620	62	(100/1000/3000)	odd vs even
Pen-Based Handwritten Digits	10992	16	(100/1000/3000)	odd vs even
Landsat Satellite	6435	36	(100/1000/3000)	$<= 3$ vs > 3
Segment	2310	19	(100/1000/1210)	$<= 3$ vs > 3
Splice	3175	60	(100/1000/2075)	

Table 7.2 shows the results of the experiments. In each column we emphasize the best result in boldface.

Experiments of Group Feature Selection (GA vs. Binary PSO) with K-Nearest Neighbor (KNN) Classifier

We repeat the experiment with K-Nearest Neighbor Classifier (Shalev-Shwartz and Ben-David [137]). In the first step, we examine the best specification of K from $\{3, 5, 7, 9, 11, 13\}$ with all features. Then we conduct the same experiment as above with the optimal K. Table 7.3 shows the results of the experiments.

Table 7.2: Experiments of Group Feature Selection (GA vs. Binary PSO)

Dataset	Feature		6×5		10×10			20×20			
			Best	Mean25	Best	Mean50	Mean100	Best	Mean50	Mean100	Mean200
Credit	24	GA	81.56 ± 1.29	79.74 ± 1.02	81.68 ± 1.41	80.90 ± 1.42	80.20 ± 1.33	81.73 ± 1.32	81.44 ± 1.31	81.25 ± 1.36	80.83 ± 1.41
		B-SPSO	**81.59** ± 1.25	**80.65** ± 1.14	**81.77** ± 1.23	**81.53** ± 1.34	**80.95** ± 1.26	**81.81** ± 1.22	81.79 ± 1.22	**81.75** ± 1.23	81.59 ± 1.27
		B-FLRPSO	81.58 ± 1.32	80.36 ± 1.25	81.72 ± 1.29	81.21 ± 1.31	80.59 ± 1.21	81.80 ± 1.29	**81.79** ± 1.25	81.74 ± 1.26	**81.60** ± 1.27
Crowd	28	GA	90.22 ± 1.07	87.55 ± 1.22	90.73 ± 1.10	89.16 ± 1.13	88.26 ± 1.20	90.95 ± 1.12	89.95 ± 0.96	89.60 ± 0.98	89.09 ± 1.01
		B-SPSO	**90.82** ± 1.08	**89.17** ± 1.13	**91.27** ± 1.09	**90.51** ± 1.07	**89.69** ± 1.07	**91.74** ± 0.87	**91.55** ± 0.85	**91.40** ± 0.87	**91.08** ± 0.89
		B-FLRPSO	90.34 ± 1.31	88.55 ± 1.16	90.94 ± 1.12	89.90 ± 1.02	88.99 ± 1.14	91.53 ± 0.90	91.12 ± 0.94	90.90 ± 0.95	90.52 ± 0.98
Drive	49	GA	76.94 ± 2.56	71.72 ± 2.76	78.69 ± 3.29	75.16 ± 2.28	73.27 ± 2.35	79.59 ± 2.97	76.60 ± 2.47	75.87 ± 2.45	74.87 ± 2.48
		B-SPSO	**78.36** ± 3.09	**74.96** ± 2.65	**80.38** ± 3.68	**78.44** ± 2.95	**76.39** ± 2.61	**82.18** ± 4.22	**81.98** ± 4.07	**81.68** ± 4.00	**80.76** ± 3.74
		B-FLRPSO	77.74 ± 2.99	73.64 ± 2.70	79.55 ± 3.23	76.28 ± 2.57	74.65 ± 2.45	81.93 ± 3.53	79.51 ± 2.96	78.74 ± 2.83	77.63 ± 2.74
German	24	GA	**72.18** ± 2.37	70.18 ± 1.77	72.58 ± 1.87	71.30 ± 1.68	70.62 ± 1.56	72.54 ± 1.86	71.95 ± 1.74	71.59 ± 1.75	71.11 ± 1.67
		B-SPSO	72.04 ± 2.44	**70.93** ± 1.94	**72.75** ± 2.06	**72.21** ± 2.16	**71.56** ± 2.03	**73.16** ± 1.96	**72.95** ± 1.71	**72.82** ± 1.66	**72.49** ± 1.61
		B-FLRPSO	72.16 ± 2.54	70.76 ± 2.10	72.74 ± 2.20	71.64 ± 1.95	70.99 ± 1.85	72.92 ± 1.85	72.63 ± 1.83	72.49 ± 1.78	72.24 ± 1.85
Letter	16	GA	71.46 ± 2.19	66.37 ± 2.00	72.29 ± 2.13	69.14 ± 2.04	67.28 ± 2.13	73.50 ± 2.33	71.30 ± 1.75	70.49 ± 1.73	69.39 ± 1.61
		B-SPSO	**72.80** ± 2.42	**69.24** ± 1.98	**74.26** ± 2.34	**71.74** ± 2.01	**70.35** ± 1.88	**74.64** ± 2.18	73.41 ± 2.04	72.65 ± 2.06	70.93 ± 2.05
		B-FLRPSO	72.17 ± 2.62	67.87 ± 2.20	73.82 ± 2.27	71.03 ± 1.90	69.06 ± 1.85	74.64 ± 2.28	**73.47** ± 1.99	**72.89** ± 1.97	**71.80** ± 1.86
Opt Digit	62	GA	94.06 ± 1.15	90.36 ± 1.41	94.75 ± 1.19	92.81 ± 1.01	91.46 ± 1.22	94.88 ± 0.95	93.78 ± 1.13	93.32 ± 1.17	92.57 ± 1.25
		B-SPSO	94.50 ± 0.89	**92.65** ± 1.32	**95.29** ± 0.96	**94.56** ± 1.22	**93.51** ± 1.14	**95.67** ± 0.91	**95.57** ± 0.92	**95.47** ± 0.92	**95.19** ± 0.94
		B-FLRPSO	**94.55** ± 1.17	91.71 ± 1.46	94.76 ± 1.06	93.19 ± 1.45	92.25 ± 1.28	95.38 ± 1.04	94.98 ± 0.91	94.75 ± 0.95	94.27 ± 1.04
Pen Digit	16	GA	94.43 ± 1.02	90.06 ± 1.79	94.80 ± 1.00	93.11 ± 1.20	91.42 ± 1.55	95.11 ± 0.91	94.18 ± 0.89	93.73 ± 0.93	92.90 ± 1.05
		B-SPSO	**94.74** ± 1.00	**92.23** ± 1.22	95.23 ± 0.95	**94.17** ± 1.00	**93.11** ± 1.14	95.27 ± 1.01	95.03 ± 0.94	94.73 ± 0.99	93.40 ± 1.17
		B-FLRPSO	94.73 ± 1.05	91.51 ± 1.43	**95.30** ± 0.93	94.08 ± 0.95	92.37 ± 1.08	**95.42** ± 0.96	**95.11** ± 0.87	**94.88** ± 0.87	**94.39** ± 0.91
Satellite	36	GA	93.02 ± 0.97	92.42 ± 1.13	93.15 ± 0.96	92.96 ± 1.00	92.69 ± 1.06	93.38 ± 0.79	93.18 ± 0.89	93.11 ± 0.92	92.98 ± 0.99
		B-SPSO	**93.17** ± 1.05	**92.70** ± 1.07	**93.29** ± 0.76	**93.11** ± 0.82	**92.86** ± 0.93	**93.51** ± 0.75	**93.45** ± 0.77	**93.41** ± 0.78	**93.33** ± 0.80
		B-FLRPSO	93.01 ± 0.95	92.49 ± 1.11	93.24 ± 0.87	92.92 ± 0.98	92.66 ± 1.06	93.45 ± 0.77	93.28 ± 0.78	93.21 ± 0.82	93.07 ± 0.88
Segment	19	GA	91.84 ± 1.67	87.19 ± 2.09	92.36 ± 1.59	90.01 ± 1.74	88.16 ± 1.93	92.61 ± 1.65	91.48 ± 1.61	90.86 ± 1.72	89.81 ± 1.81
		B-SPSO	**92.16** ± 1.58	**89.77** ± 1.71	**92.54** ± 1.67	**91.59** ± 1.68	**90.18** ± 1.66	92.80 ± 1.53	**92.56** ± 1.53	92.30 ± 1.53	91.07 ± 1.62
		B-FLRPSO	91.77 ± 1.69	88.54 ± 2.01	92.52 ± 1.75	91.22 ± 1.75	89.50 ± 1.87	**92.88** ± 1.61	92.54 ± 1.54	**92.35** ± 1.51	**91.88** ± 1.52
Splice	60	GA	78.66 ± 2.13	70.39 ± 2.03	79.54 ± 1.70	75.72 ± 1.59	72.66 ± 1.80	80.03 ± 1.52	77.96 ± 1.40	76.83 ± 1.47	75.06 ± 1.56
		B-SPSO	**79.55** ± 1.91	**75.36** ± 1.94	**81.64** ± 1.65	**80.01** ± 1.90	**77.55** ± 1.43	**82.35** ± 1.63	**82.11** ± 1.50	**81.89** ± 1.43	**81.35** ± 1.38
		B-FLRPSO	78.96 ± 2.06	73.50 ± 2.58	80.41 ± 1.66	77.56 ± 2.34	74.99 ± 1.60	81.79 ± 1.33	80.75 ± 1.31	80.22 ± 1.35	79.15 ± 1.49

Results: In Table 7.2, B-SPSO and B-FLRPSO work better than GA in most cases and the performance of B-SPSO is better than that of B-FLRPSO. The results in Table 7.2 are uniformly better than those in Table 7.3. However the relative performance among those three methods does not change so much. In these tables B-SPSO shows the best results. Therefore we choose B-SPSO for the next experiment.

Experiments of Group Feature Selection (SFS, SBS vs. Binary PSO)

We compare the performance of sequential selection methods (sequential forward selection, sequential backward selection) and B-SPSO. We use Gaussian kernel and the same specification for the hyper-parameters and α-values as in Chapter 3 and datasets in Table 7.2.

1. **SFS**: SFS in Section 2.10.2:
   This method start with a set F_1 which consists of single feature subsets.

$$F_1 = \{\{f_1\}, \ldots, \{f_d\}\}$$

Table 7.3: Experiments of Group Feature Selection (GA vs. Binary PSO) with KNN Classifier

Dataset	Feature		6×5		10×10			20×20			
			Best	Mean25	Best	Mean50	Mean100	Best	Mean50	Mean100	Mean200
Credit	24	GA	80.80 ± 0.95	78.98 ± 0.98	80.90 ± 1.09	79.88 ± 0.85	79.28 ± 0.85	81.32 ± 0.91	80.47 ± 0.82	80.23 ± 0.84	79.85 ± 0.84
		B-SPSO	**81.33** ± 1.20	**80.29** ± 0.92	**81.75** ± 0.98	**81.36** ± 0.85	**80.76** ± 0.81	**82.16** ± 0.81	**82.01** ± 0.74	**81.88** ± 0.70	**81.55** ± 0.72
		B-FLRPSO	80.81 ± 1.19	79.36 ± 0.83	81.03 ± 0.92	80.24 ± 0.83	79.59 ± 0.78	81.52 ± 0.97	81.14 ± 0.89	80.94 ± 0.84	80.64 ± 0.80
Crowd	28	GA	87.48 ± 1.59	84.01 ± 1.53	87.90 ± 1.51	85.79 ± 1.50	84.65 ± 1.60	88.36 ± 1.32	86.97 ± 1.34	86.49 ± 1.38	85.81 ± 1.42
		B-SPSO	87.61 ± 1.46	**85.67** ± 1.58	**88.79** ± 1.32	**87.76** ± 1.41	**86.65** ± 1.52	89.09 ± 1.32	**88.85** ± 1.25	**88.65** ± 1.24	**88.23** ± 1.26
		B-FLRPSO	**87.74** ± 1.78	85.09 ± 1.78	88.20 ± 1.35	86.88 ± 1.25	85.66 ± 1.31	**89.10** ± 1.22	88.41 ± 1.10	88.13 ± 1.13	87.64 ± 1.17
Drive	49	GA	73.29 ± 3.90	67.14 ± 2.18	76.46 ± 4.68	69.52 ± 1.80	68.04 ± 1.88	79.08 ± 4.08	71.00 ± 1.90	70.06 ± 1.86	69.03 ± 1.89
		B-SPSO	73.38 ± 4.87	**69.73** ± 2.43	**79.75** ± 6.11	**76.08** ± 4.56	**72.64** ± 2.60	**85.49** ± 5.51	**84.86** ± 5.28	**84.07** ± 5.09	**81.79** ± 4.69
		B-FLRPSO	**73.82** ± 4.26	68.36 ± 2.22	77.88 ± 5.08	70.73 ± 2.12	69.28 ± 1.93	83.58 ± 3.70	78.46 ± 4.14	75.90 ± 3.56	73.28 ± 2.70
German	24	GA	70.04 ± 2.13	68.69 ± 1.62	70.32 ± 1.94	69.59 ± 1.41	69.17 ± 1.47	70.76 ± 2.60	70.09 ± 1.47	69.86 ± 1.44	69.57 ± 1.42
		B-SPSO	70.46 ± 2.08	**69.78** ± 1.58	70.80 ± 1.92	**70.63** ± 1.44	**70.17** ± 1.41	**71.56** ± 1.64	**71.32** ± 1.48	**71.19** ± 1.42	**70.92** ± 1.34
		B-FLRPSO	**70.70** ± 2.07	69.27 ± 1.38	**71.07** ± 2.13	69.93 ± 1.40	69.41 ± 1.43	71.24 ± 2.00	70.78 ± 1.42	70.58 ± 1.39	70.28 ± 1.36
Letter	16	GA	69.30 ± 2.43	64.00 ± 1.96	69.81 ± 2.52	66.73 ± 1.71	65.03 ± 1.72	71.26 ± 1.97	68.49 ± 1.68	67.66 ± 1.61	66.58 ± 1.57
		B-SPSO	**70.59** ± 2.87	**66.56** ± 2.27	**72.44** ± 2.66	**69.29** ± 2.07	**67.92** ± 1.88	**73.18** ± 2.44	**71.38** ± 2.26	70.39 ± 2.18	68.36 ± 1.92
		B-FLRPSO	69.42 ± 2.41	65.20 ± 2.07	71.84 ± 2.47	68.35 ± 1.65	66.55 ± 1.57	72.66 ± 2.45	71.12 ± 2.06	**70.40** ± 2.00	**69.19** ± 1.79
Opt Digit	62	GA	93.32 ± 1.29	89.62 ± 1.28	93.49 ± 1.04	91.73 ± 1.13	90.39 ± 1.32	94.27 ± 0.95	93.07 ± 0.90	92.63 ± 0.96	91.93 ± 1.02
		B-SPSO	**93.67** ± 0.99	**91.71** ± 0.91	94.06 ± 1.00	**93.16** ± 1.26	**92.05** ± 0.92	**94.61** ± 0.91	**94.30** ± 0.73	**94.13** ± 0.71	**93.77** ± 0.72
		B-FLRPSO	93.61 ± 1.36	91.15 ± 1.41	**94.12** ± 1.10	92.48 ± 1.51	91.52 ± 1.12	94.46 ± 1.00	94.17 ± 0.85	93.90 ± 0.88	93.38 ± 0.98
Pen Digit	16	GA	93.40 ± 0.94	89.54 ± 1.70	93.77 ± 1.04	91.82 ± 1.19	90.35 ± 1.31	94.08 ± 1.03	93.08 ± 1.02	92.65 ± 1.09	91.92 ± 1.15
		B-SPSO	**93.77** ± 1.25	**91.47** ± 1.51	**94.22** ± 1.10	**93.14** ± 1.20	**92.11** ± 1.24	94.27 ± 1.09	94.04 ± 1.09	93.70 ± 1.19	92.51 ± 1.30
		B-FLRPSO	93.74 ± 1.26	90.79 ± 1.36	94.05 ± 0.95	93.05 ± 1.05	91.54 ± 1.17	**94.39** ± 0.93	**94.14** ± 0.85	**93.89** ± 0.87	**93.37** ± 0.94
Satellite	36	GA	**92.29** ± 0.91	91.88 ± 0.86	**92.40** ± 0.75	92.11 ± 0.82	91.95 ± 0.89	92.34 ± 0.78	92.23 ± 0.81	92.18 ± 0.82	92.11 ± 0.83
		B-SPSO	92.26 ± 0.93	**92.01** ± 0.92	92.32 ± 0.83	**92.19** ± 0.82	**92.05** ± 0.86	92.50 ± 0.82	**92.43** ± 0.81	**92.40** ± 0.82	**92.34** ± 0.82
		B-FLRPSO	92.28 ± 0.83	91.93 ± 0.91	92.35 ± 0.80	92.16 ± 0.85	92.04 ± 0.86	**92.55** ± 0.77	92.38 ± 0.83	92.34 ± 0.83	92.26 ± 0.84
Segment	19	GA	90.98 ± 1.85	87.37 ± 1.66	91.41 ± 1.76	89.31 ± 1.59	88.03 ± 1.58	91.85 ± 1.78	90.63 ± 1.72	90.14 ± 1.75	89.36 ± 1.77
		B-SPSO	**91.32** ± 1.89	**88.70** ± 1.76	**92.02** ± 1.50	**90.84** ± 1.43	**89.69** ± 1.55	**92.45** ± 1.56	**92.01** ± 1.63	91.57 ± 1.64	90.27 ± 1.69
		B-FLRPSO	91.18 ± 1.83	88.42 ± 1.84	91.82 ± 2.06	90.49 ± 1.70	89.11 ± 1.62	92.38 ± 1.83	91.87 ± 1.79	**91.62** ± 1.78	**91.09** ± 1.79
Splice	60	GA	71.63 ± 3.26	64.47 ± 2.54	72.95 ± 3.04	68.37 ± 2.61	65.68 ± 2.43	73.73 ± 2.66	71.33 ± 2.43	70.22 ± 2.37	68.44 ± 2.21
		B-SPSO	71.27 ± 3.76	**67.64** ± 3.68	73.53 ± 3.15	**71.91** ± 3.15	**69.54** ± 2.95	**76.07** ± 2.71	**75.47** ± 2.57	**75.06** ± 2.59	**74.20** ± 2.68
		B-FLRPSO	**72.51** ± 2.73	67.44 ± 2.67	**73.76** ± 2.61	69.91 ± 3.15	68.13 ± 2.56	75.84 ± 2.73	74.20 ± 2.34	73.57 ± 2.35	72.39 ± 2.38

We evaluate each $\{f_i\}$ by running SVM. Let $B_1 = \{f_{i^*}\}$ be the best subset. In the next round, we construct a set F_2 which consists of feature subsets $\{f_{i^*}, f_j\}$ for $j \neq i^*$.

$$F_2 = \{\{f_{i^*}, f_j\} | j \neq i^*\}$$

We evaluate each subset in F_2 by running SVM. Let $B_2 = \{f_{i^*}, f_{j^*}\}$ be the best subset. If the fitness value of B_2 is better than or equal to that of B_1, we continue the process until the fitness value of B_{k+1} is worse than that of B_k or $k \geq d$.

2. **SBS**: SBS in Section 2.10.2:
   This method start with a set B_0 which consists of all features.

$$B_0 = \{f_1, \ldots, f_d\}$$

We construct a set F_1 which consists of feature subsets $B_0 \setminus \{f_i\}$ for $i = 1, \ldots, d$.

$$F_1 = \{(B_0 \setminus \{f_1\}), \ldots, (B_0 \setminus \{f_d\})\}$$

Let $B_1 = B_0 \setminus \{f_{i^*}\}$ be the best subset in F_1. If the fitness value of B_1 is better than or equal to that of B_0, we continuer the process. In the next round, we construct a set F_2:

$$F_2 = \{B_1 \setminus \{f_j\} | j \neq i^*\}$$

Let $B_2 = B_1 \setminus \{f_{j^*}\}$ be the best subset of F_2. If the fitness value of B_2 is better than or equal to that of B_1, we continuer the process until the fitness value of B_{k+1} is worse than that of B_k or $k \geq d$.

3. **B-SPSO** in Section 4.3.3:
   We set $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$. We also set the parameter ξ so that ξ is 0.08 at the first iteration and lineally decreases to 0.01 at the last iteration. We examine four patterns of population size and the number of generations for B-SPSO.

   - B-SPSO-5: population size = 5 and number of generations = 5
   - B-SPSO-10: population size = 10 and number of generations = 10
   - B-SPSO-20: population size = 20 and number of generations = 20
   - B-SPSO-30: population size = 30 and number of generations = 30

We report the following results using test data.

- Best: evaluation for the best feature subset estimation
- Mean50(25): mean of the test evaluations for the top 25 group feature selection for B-SPSO-5
  mean of the test evaluations for the top 50 group feature selection for B-SPSO-10, B-SPSO-20 and B-SPSO-30
- Mean100: mean of the test evaluations for the top 100 group feature selection

Results and Discussions

Table 7.4 shows the results of the experiments. The main comparison is Backward versus B-SPSO-20. The best result in each row is emphasized in boldface. If the results between Backward and B-SPSO-20 are significantly different (p-values < 0.05 for the two-tailed paired t-test), (*) is shown beside the number in the column with the better results.

In Table 7.4, the results of SBS are better than those of SFS in most cases. For the main comparison, the results of B-SPSO-20 are significantly better than SBS for 12 rows and significantly worse than SBS for 4 rows out of 30 rows.

Table 7.4: Experiments of Group Feature Selection (Forward Selection, Backward Elimination vs. B-SPSO)

Dataset	Feature		**SFS**	**SBS**	**B-SPSO-5**	**B-SPSO-10**	**B-SPSO-20**	**B-SPSO-30**
Credit	24	Max	**82.00** $\pm$ 1.52	81.33 $\pm$ 1.27	81.64 $\pm$ 1.39	81.71 $\pm$ 1.47	81.71* $\pm$ 1.32	81.84 $\pm$ 1.31
		Mean50(25)	**81.88** $\pm$ 1.46	81.32 $\pm$ 1.27	80.76 $\pm$ 1.30	81.47 $\pm$ 1.42	81.72* $\pm$ 1.36	81.82 $\pm$ 1.30
		Mean100	81.70 $\pm$ 1.47	81.29 $\pm$ 1.27		80.85 $\pm$ 1.21	81.70* $\pm$ 1.38	**81.81** $\pm$ 1.29
			(206.1)	(168.6)	(25)	(100)	(400)	(900)
Crowd	28	Max	90.07 $\pm$ 1.44	**91.56** $\pm$ 0.98	90.36 $\pm$ 1.32	90.97 $\pm$ 1.19	91.46 $\pm$ 1.07	91.33 $\pm$ 1.05
		Mean50(25)	89.73 $\pm$ 1.35	**91.49*** $\pm$ 1.02	88.69 $\pm$ 1.32	90.25 $\pm$ 1.23	91.23 $\pm$ 1.07	91.30 $\pm$ 0.99
		Mean100	89.18 $\pm$ 1.44	**91.42*** $\pm$ 1.04		89.37 $\pm$ 1.29	91.11 $\pm$ 1.09	91.23 $\pm$ 1.00
			(305.5)	(134.7)	(25)	(100)	(400)	(900)
Drive	49	Max	80.67 $\pm$ 8.89	77.83 $\pm$ 2.61	77.15 $\pm$ 3.03	78.61 $\pm$ 2.45	**81.20*** $\pm$ 3.44	80.74 $\pm$ 3.27
		Mean50(25)	80.65 $\pm$ 8.89	77.81 $\pm$ 2.62	74.29 $\pm$ 2.67	77.41 $\pm$ 2.36	**80.99*** $\pm$ 3.16	80.72 $\pm$ 3.26
		Mean100	80.62 $\pm$ 8.90	77.77 $\pm$ 2.63		75.48 $\pm$ 2.45	**80.77*** $\pm$ 2.98	80.70 $\pm$ 3.21
			(673.2)	(733.5)	(25)	(100)	(400)	(900)
German	24	Max	**72.79** $\pm$ 2.13	72.55 $\pm$ 1.98	72.22 $\pm$ 2.52	72.52 $\pm$ 2.25	72.47 $\pm$ 2.53	72.64 $\pm$ 2.09
		Mean50(25)	72.47 $\pm$ 1.97	72.39 $\pm$ 1.94	71.03 $\pm$ 1.86	72.14 $\pm$ 2.06	72.50 $\pm$ 2.20	**72.68** $\pm$ 1.94
		Mean100	72.09 $\pm$ 1.91	72.21 $\pm$ 1.88		71.52 $\pm$ 1.88	72.38 $\pm$ 2.09	**72.66** $\pm$ 1.95
			(213.8)	(134.8)	(25)	(100)	(400)	(900)
Letter	16	Max	71.41 $\pm$ 3.18	74.26 $\pm$ 2.45	72.65 $\pm$ 2.51	73.81 $\pm$ 2.60	74.33 $\pm$ 2.43	**74.54** $\pm$ 2.51
		Mean50(25)	67.77 $\pm$ 4.11	72.96 $\pm$ 2.19	68.74 $\pm$ 2.08	71.41 $\pm$ 2.04	73.31* $\pm$ 2.25	**73.70** $\pm$ 2.27
		Mean100	65.09 $\pm$ 3.06	72.57 $\pm$ 2.09		70.11 $\pm$ 1.91	72.60 $\pm$ 2.17	**73.24** $\pm$ 2.20
			(87.3)	(60.7)	(25)	(100)	(400)	(900)
Opt Digit	62	Max	93.89 $\pm$ 1.83	**95.62** $\pm$ 1.02	94.76 $\pm$ 1.13	94.94 $\pm$ 1.29	95.57 $\pm$ 1.12	95.45 $\pm$ 1.15
		Mean50(25)	93.89 $\pm$ 1.82	**95.62** $\pm$ 1.03	92.72 $\pm$ 1.36	94.55 $\pm$ 1.32	95.51 $\pm$ 1.09	95.45 $\pm$ 1.14
		Mean100	93.87 $\pm$ 1.84	**95.62*** $\pm$ 1.03		93.35 $\pm$ 1.39	95.45 $\pm$ 1.09	95.43 $\pm$ 1.14
			(1109.4)	(863.9)	(25)	(100)	(400)	(900)
Pen Digit	16	Max	94.92 $\pm$ 0.97	95.51 $\pm$ 0.93	94.79 $\pm$ 1.00	95.25 $\pm$ 0.96	95.50 $\pm$ 0.85	**95.52** $\pm$ 0.90
		Mean50(25)	93.46 $\pm$ 1.32	95.09 $\pm$ 0.91	91.97 $\pm$ 1.57	94.05 $\pm$ 1.08	95.12 $\pm$ 0.80	**95.23** $\pm$ 0.82
		Mean100	88.72 $\pm$ 2.92	94.98* $\pm$ 0.93		92.95 $\pm$ 1.22	94.77 $\pm$ 0.84	**95.04** $\pm$ 0.83
			(120.0)	(55.1)	(25)	(100)	(400)	(900)
Satellite	36	Max	92.55 $\pm$ 1.35	93.54 $\pm$ 0.61	93.27 $\pm$ 0.64	93.45 $\pm$ 0.63	93.53 $\pm$ 0.67	**93.62** $\pm$ 0.58
		Mean50(25)	92.37 $\pm$ 1.44	93.52 $\pm$ 0.61	92.93 $\pm$ 0.71	93.31 $\pm$ 0.64	93.46 $\pm$ 0.63	**93.57** $\pm$ 0.55
		Mean100	92.06 $\pm$ 1.86	93.51 $\pm$ 0.61		93.09 $\pm$ 0.67	93.44 $\pm$ 0.63	**93.55** $\pm$ 0.54
			(311.5)	(257.5)	(25)	(100)	(400)	(900)
Segment	19	Max	92.65 $\pm$ 2.46	93.04 $\pm$ 1.50	92.39 $\pm$ 1.69	93.02 $\pm$ 1.49	93.07 $\pm$ 1.54	**93.28** $\pm$ 1.52
		Mean50(25)	91.89 $\pm$ 2.44	92.69 $\pm$ 1.45	89.97 $\pm$ 1.70	91.87 $\pm$ 1.33	92.91* $\pm$ 1.49	**93.10** $\pm$ 1.50
		Mean100	88.98 $\pm$ 3.42	92.24 $\pm$ 1.35		90.55 $\pm$ 1.37	92.61* $\pm$ 1.45	**92.94** $\pm$ 1.44
			(132.1)	(95.3)	(25)	(100)	(400)	(900)
Splice	60	Max	79.84 $\pm$ 2.70	80.84 $\pm$ 1.41	80.00 $\pm$ 1.92	81.56 $\pm$ 1.41	82.57* $\pm$ 1.39	**83.20** $\pm$ 1.46
		Mean50(25)	79.17 $\pm$ 3.21	80.77 $\pm$ 1.38	75.54 $\pm$ 1.87	80.33 $\pm$ 1.39	82.36* $\pm$ 1.40	**83.08** $\pm$ 1.40
		Mean100	78.68 $\pm$ 3.94	80.72 $\pm$ 1.39		77.62 $\pm$ 1.43	82.17* $\pm$ 1.35	**82.98** $\pm$ 1.38
			(550.1)	(538.9)	(25)	(100)	(400)	(900)

Experiments of Embedded Method vs. Wrapper Method:

Next we compare the performance of H-MKL which represents an embedded method and B-SPSO as a wrapper method. We use Gaussian kernel and the same specification

for the hyper-parameters and α-values as in Chapter 3.

1. **H-MKL** the heuristic method based on WR-MKL in Section 7.3.1:
   We set $n_{pw} = 8$ in Section 7.3.1.

2. **B-SPSO** in Section 4.3.3:
   We set $(\omega, c_1, c_2) = (0.729, 1.49, 1.49)$. We also set the parameter ξ so that ξ is 0.08 at the first iteration and lineally decreases to 0.01 at the last iteration. We examine four patterns of population size and the number of generations for B-SPSO.

   - B-SPSO-5: population size = 5 and number of generations = 5
   - B-SPSO-10: population size = 10 and number of generations = 10
   - B-SPSO-20: population size = 20 and number of generations = 20
   - B-SPSO-30: population size = 30 and number of generations = 30

   We set $N_F = 200$ in Section 7.3.1 and report the following results using test data.

- Best: evaluation for the best feature subset estimation
- Mean50(25): mean of the test evaluations for the top 25 group feature selection for B-SPSO-5
  mean of the test evaluations for the top 50 group feature selection for B-SPSO-10, B-SPSO-20 and B-SPSO-30
- Mean100: mean of the test evaluations for the top 100 group feature selection
- Mean200: mean of the test evaluations for the top 200 group feature selection

Results and Discussions

Table 7.5 shows the results of the experiments. The main comparison is H-MKL versus B-SPSO-20. The best result between the two is emphasized in boldface. If the results between H-MKL and B-SPSO-20 are significantly different (p-values < 0.05 for the two-tailed paired t-test), (*) is shown beside the number in the column with the better results. We also compare the performance of H-MKL with B-SPSO-5. If the results between H-MKL and B-SPSO-5 are significantly different (p-values < 0.05 for the two-tailed paired t-test), ($^{(*)}$) is shown beside the number in the column with the better results.

In Table 7.5, the results of B-SPSO-20 are significantly better than H-MKL for all rows. The total numbers of the H-MKL runs and the SVM runs for the binary PSO

Table 7.5: Experiments of Group Feature Selection (H-MKL vs. B-SPSO)

Dataset	Feature		**H-MKL**	**B-SPSO-5**	**B-SPSO-10**	**B-SPSO-20**	**B-SPSO-30**
Credit	24	Best	79.30 ± 1.47	$79.68^{(*)} \pm 1.68$	79.83 ± 1.43	$\mathbf{79.93}^{*} \pm 1.55$	80.04 ± 1.54
		Mean50(25)	79.06 ± 1.41	79.06 ± 1.44	79.63 ± 1.46	$\mathbf{79.91}^{*} \pm 1.55$	80.02 ± 1.54
		Mean100	79.05 ± 1.41		79.24 ± 1.35	$\mathbf{79.88}^{*} \pm 1.55$	80.01 ± 1.53
		Mean200	79.06 ± 1.44			$\mathbf{79.71}^{*} \pm 1.55$	79.97 ± 1.53
Crowd	28	Best	91.13 ± 2.26	90.83 ± 1.33	91.33 ± 1.01	$\mathbf{91.76} \pm 0.80$	91.72 ± 0.97
		Mean50(25)	89.57 ± 1.79	89.42 ± 1.31	90.78 ± 1.08	$\mathbf{91.65}^{*} \pm 0.85$	91.70 ± 0.94
		Mean100	89.47 ± 1.75		89.98 ± 1.08	$\mathbf{91.53}^{*} \pm 0.86$	91.64 ± 0.93
		Mean200	89.45 ± 1.66			$\mathbf{91.24}^{*} \pm 0.88$	91.53 ± 0.94
Drive	49	Best	75.78 ± 2.31	$78.33^{(*)} \pm 3.02$	79.88 ± 2.39	$\mathbf{81.70}^{*} \pm 4.18$	82.33 ± 3.87
		Mean50(25)	72.96 ± 4.74	$75.13^{(*)} \pm 2.05$	78.34 ± 2.29	$\mathbf{81.46}^{*} \pm 3.95$	82.32 ± 3.85
		Mean100	73.18 ± 4.39		76.49 ± 1.98	$\mathbf{81.09}^{*} \pm 3.75$	82.25 ± 3.83
		Mean200	73.24 ± 4.24			$\mathbf{80.19}^{*} \pm 3.27$	82.03 ± 3.74
German	24	Best	71.58 ± 2.28	71.83 ± 2.25	72.60 ± 2.21	$\mathbf{72.52}^{*} \pm 2.13$	72.60 ± 2.10
		Mean50(25)	$71.31^{(*)} \pm 1.74$	70.84 ± 1.67	72.01 ± 1.85	$\mathbf{72.36}^{*} \pm 1.99$	72.53 ± 2.01
		Mean100	71.26 ± 1.71		71.41 ± 1.63	$\mathbf{72.24}^{*} \pm 1.93$	72.48 ± 2.01
		Mean200	71.23 ± 1.69			$\mathbf{71.94}^{*} \pm 1.83$	72.39 ± 1.95
Letter	16	Best	71.02 ± 4.21	$72.15^{(*)} \pm 2.49$	73.58 ± 2.50	$\mathbf{73.95}^{*} \pm 2.49$	74.17 ± 2.51
		Mean50(25)	67.96 ± 2.52	68.58 ± 2.00	70.97 ± 2.31	$\mathbf{72.91}^{*} \pm 2.20$	73.31 ± 2.21
		Mean100	67.85 ± 2.44		69.61 ± 2.04	$\mathbf{72.14}^{*} \pm 2.07$	72.84 ± 2.14
		Mean200	67.79 ± 2.43			$\mathbf{70.44}^{*} \pm 1.84$	71.93 ± 2.03
Opt Digit	62	Best	92.83 ± 3.99	$94.58^{(*)} \pm 1.09$	95.31 ± 0.96	$\mathbf{95.74}^{*} \pm 0.90$	95.93 ± 0.87
		Mean50(25)	91.39 ± 3.53	$92.81^{(*)} \pm 1.24$	94.76 ± 1.04	$\mathbf{95.68}^{*} \pm 0.91$	95.87 ± 0.87
		Mean100	91.51 ± 3.33		93.64 ± 1.09	$\mathbf{95.59}^{*} \pm 0.92$	95.85 ± 0.87
		Mean200	91.58 ± 3.27			$\mathbf{95.35}^{*} \pm 0.95$	95.79 ± 0.87
Pen Digit	16	Best	94.91 ± 2.03	95.18 ± 0.96	95.50 ± 0.99	$\mathbf{95.69}^{*} \pm 1.00$	95.73 ± 0.96
		Mean50(25)	92.01 ± 2.15	92.26 ± 1.41	94.23 ± 1.05	$\mathbf{95.34}^{*} \pm 0.95$	95.44 ± 0.90
		Mean100	91.93 ± 1.94		93.07 ± 1.24	$\mathbf{95.01}^{*} \pm 0.94$	95.23 ± 0.92
		Mean200	91.89 ± 1.78			$\mathbf{93.63}^{*} \pm 1.07$	94.75 ± 0.99
Satellite	36	Best	92.26 ± 2.35	$93.36^{(*)} \pm 0.60$	93.52 ± 0.63	$\mathbf{93.51}^{*} \pm 0.59$	93.53 ± 0.68
		Mean50(25)	91.86 ± 1.95	$92.96^{(*)} \pm 0.74$	93.35 ± 0.62	$\mathbf{93.51}^{*} \pm 0.57$	93.50 ± 0.62
		Mean100	91.97 ± 1.77		93.10 ± 0.67	$\mathbf{93.49}^{*} \pm 0.57$	93.49 ± 0.61
		Mean200	92.04 ± 1.70			$\mathbf{93.43}^{*} \pm 0.57$	93.47 ± 0.60
Segment	19	Best	91.21 ± 2.60	$92.34^{(*)} \pm 1.56$	92.79 ± 1.51	$\mathbf{92.80}^{*} \pm 1.58$	92.80 ± 1.55
		Mean50(25)	90.12 ± 2.70	90.01 ± 1.90	91.92 ± 1.61	$\mathbf{92.69}^{*} \pm 1.59$	92.70 ± 1.57
		Mean100	90.16 ± 2.70		90.61 ± 1.65	$\mathbf{92.45}^{*} \pm 1.57$	92.58 ± 1.56
		Mean200	90.09 ± 2.73			$\mathbf{91.30}^{*} \pm 1.67$	92.25 ± 1.57
Splice	60	Best	76.01 ± 3.88	$79.51^{(*)} \pm 2.27$	81.25 ± 1.42	$\mathbf{82.30}^{*} \pm 1.57$	82.84 ± 1.22
		Mean50(25)	75.15 ± 2.55	75.45 ± 2.15	80.14 ± 1.45	$\mathbf{82.12}^{*} \pm 1.40$	82.77 ± 1.16
		Mean100	75.06 ± 2.43		77.62 ± 1.44	$\mathbf{81.89}^{*} \pm 1.38$	82.68 ± 1.18
		Mean200	75.08 ± 2.44			$\mathbf{81.38}^{*} \pm 1.37$	82.52 ± 1.18

are $d - 7$ and $20 \times 20 = 200$, respectively. The difference is not small and it may not be a fair comparison. The average number of MKL runs for H-MKL is 25.4. The comparison of H-MKL with B-SPSO-5 also shows that B-SPSO outperforms H-MKL. In general the direct measurements by wrapper methods are more accurate than indirect measurements by embedded methods.

We will use B-SPSO as the group feature selection method in the experiments of the optimal kernel construction.

7.4.2 Experiments for Optimal Kernel Construction

Now we conduct experiments for Optimal Kernel Construction.

Gaussian kernels and Polynomial kernels in Section 2.3.3 are used as the basis kernels. Table 7.6 shows the ranges and step sizes of the hyper-parameters in the basic kernels.

Table 7.6: Settings of hyper-parameters in kernels

Kernel	Parameter	MKL		KC	
		Range	Step Size	Range	Step Size
$\exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{\sigma^2}\right)$	$\log \sigma^2$	$[-8, 10]$	2	$[-8, 10]$	1
$(a\langle \mathbf{x}_i, \mathbf{x}_j\rangle + b)^c$	$\log a$	$[0, 1]$	1	$[-2, 2]$	1
	b	$[1, 1]$	1	$[0, 1]$	1
	c	$[1, 7]$	1	$[1, 7]$	1

Note that we set the range of hyper-parameters of polynomial kernel smaller than Table 3.1. The reason is that for the KC subsystem those kernels are used as the basic kernels and combined by $+$, $\times$ and $\exp$ operation in Section 7.3.2. Therefore as the ingredient of the operations we prepare only small pieces of kernels. For MKL we also try to keep the number of kernels to be small, since a large number of kernels often leads to a deterioration of prediction accuracy.

We use WR-MKL in Chapter 6 as the MKL solver for binary classification. We use UFO-MKL in Section 6.2.1 as the SVM solver and the MKL solver for multi-class classification. The reason for this is that since UFO-MKL is a SGD-based method we also use the same method to solve SVM for a fair comparison. Table 7.7 shows the

ranges and step sizes of the hyper-parameters in the UFO method:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \frac{1}{2} \|\mathbf{w}\|_{2,p}^2 + \alpha \|\mathbf{w}\|_{2,1} + \frac{1}{\lambda n} \sum_{i=1}^{n} \sum_{i=1}^{n} \ell\left(\mathbf{w}, \Phi(\mathbf{x}_i), y_i\right) \quad ,$$

where p is set to $\frac{2 \log m}{2 \log m - 1}$. We set those ranges so that they cover the settings of hyper-parameters in the experiments in (Orabona et al. [117]). The matlab code of UFO-MKL is obtained from (Orabona [116]).

Table 7.7: Settings of hyper-parameters in UFO Method

Parameter	SVM		MKL	
	Range	Step Size	Range	Step Size
$\log_{10} C$	$[-3, 6]$	3	$[-3, 6]$	3
$\log_{10} \alpha$	$[-5, -1]$	2	$[-5, -1]$	2

where $C = \frac{1}{\lambda n}$. We compare the prediction accuracy of the following six methods:

1. **SVM**: SVM with fine tuned parameters
   The settings for hyper-parameters and α is the same as the SEB method with Gaussian kernels in Chapter 3.

2. **MKL 1**: MKL with a combination of the polynomial and Gaussian kernels with all the features. The parameters of the polynomial and Gaussian kernels are set as in the "MKL" column in Table 7.6.

3. **MKL 2**: MKL with a combination of Polynomial kernels and Gaussian kernels which correspond to each single feature and all the features. If a dataset has d features, the MKL has a combination of $d + 1$ (subsets of features) $\times$ 24 kernels (14 Polynomial kernels and 10 Gaussian kernels as described above).

4. **FS**: FS subsystem only
   We use B-SPSO method with both the number of iterations and the size of the population $= 20$.

5. **KC**: KC subsystem only
   We use the two types of basic kernels; Gaussian kernels and Polynomial kernels in Section 2.3.3. The parameters of those kernels are specified as in the "KC" column in Table 7.6.

6. **FS-KC**: FS-KC subsystem
   For the variable terminals of the KC subsystem, we set $n_{FS} = 30$ and $n_{KC} = 30$.

For GP, we use GPLAB (Silva and Almeida [142]). Table 7.8 gives a summary of the settings in GPLAB, which are explained in Section 2.7.

Table 7.8: Summary of settings in GPLAB

Parameter	Value
Population size	50
Number of Generations	30
Initialization	Ramped Half-and-Half
Maximum level of Node Depth	6
Maximum level for initialization	3
Selection for Reproduction	roulette
Elitism	halfelitism
Reproduction rate	0.1
Crossover/Mutation rate	0.7/0.3

We use benchmark datasets in Table 1.2 with training $= 100$ for binary classification and benchmark datasets in Table 1.3 for multi-class classification. Experiments are repeated 50 times for each dataset.

Results and Discussions

In Table 7.9, the binary classification is carried out for the 17 datasets. In Table 7.10, the multi-class classification is conducted for 10 datasets. The main comparison is SVM versus (FC, KC and FS-KC). The best result in each row is emphasized in boldface. The numbers in the parenthesis are the p-values of the paired t-test for the null hypothesis that the prediction accuracy of the corresponding methods are equal. We also compare the performance of KC and FS-KC using the paired t-test. If the results between the two methods are significantly different (p-value < 0.05), (*) is shown beside the number in the column of the better result.

Table 7.9 and Table 7.10 show that the prediction accuracy of FS-KC is significantly better than SVM in all datasets and in most cases also outperforms KC. In Table 7.9, the results of FS-KC are significantly better than those of KC for 13 datasets out of 17. In

Table 7.9: Prediction accuracy for binary classification

Dataset	SVM	MKL 1	MKL 2	FS	KC	FS-KC
Banknote	99.49 ± 0.41	98.99 ± 0.70	98.74 ± 0.91	99.47 ± 0.42 ($p = 0.335$)	$\mathbf{99.62} \pm 0.52$ ($p < 0.001$)	99.61 ± 0.44 ($p < 0.001$)
Credit	78.63 ± 1.24	$\mathbf{80.32} \pm 1.66$	79.57 ± 3.55	79.34 ± 1.59 ($p < 0.001$)	78.81 ± 1.21 ($p < 0.001$)	$79.38^{*} \pm 1.55$ ($p < 0.001$)
Crowd-sourced	91.66 ± 0.87	90.49 ± 2.13	89.74 ± 1.49	91.66 ± 0.84 ($p = 0.999$)	91.87 ± 1.02 ($p < 0.001$)	$\mathbf{92.36}^{*} \pm 0.83$ ($p < 0.001$)
Drive	75.58 ± 2.07	72.30 ± 2.64	61.04 ± 11.18	82.09 ± 3.76 ($p < 0.001$)	76.20 ± 2.06 ($p < 0.001$)	$\mathbf{84.55}^{*} \pm 4.46$ ($p < 0.001$)
German	71.78 ± 2.23	70.95 ± 1.83	72.10 ± 2.22	72.55 ± 2.71 ($p = 0.017$)	73.98 ± 2.08 ($p < 0.001$)	$\mathbf{75.29}^{*} \pm 2.00$ ($p < 0.001$)
Letter	72.50 ± 1.97	70.00 ± 2.90	71.22 ± 3.32	74.81 ± 2.44 ($p < 0.001$)	74.08 ± 1.95 ($p < 0.001$)	$\mathbf{78.14}^{*} \pm 2.21$ ($p < 0.001$)
MAGIC	81.12 ± 1.21	80.53 ± 1.51	80.03 ± 1.99	81.53 ± 1.14 ($p < 0.001$)	81.53 ± 1.19 ($p < 0.001$)	$\mathbf{81.74} \pm 1.29$ ($p < 0.001$)
Occupancy	98.74 ± 0.34	97.63 ± 1.12	98.44 ± 0.66	98.76 ± 0.33 ($p = 0.032$)	98.81 ± 0.32 ($p < 0.001$)	$\mathbf{98.94}^{*} \pm 0.20$ ($p < 0.001$)
Opt Digit	95.54 ± 1.90	95.11 ± 1.37	89.81 ± 2.40	96.36 ± 1.36 ($p < 0.001$)	95.73 ± 1.32 ($p < 0.001$)	$\mathbf{96.55}^{*} \pm 0.79$ ($p < 0.001$)
Page Blocks	93.92 ± 0.86	92.14 ± 0.90	92.65 ± 1.94	94.02 ± 0.92 ($p = 0.099$)	94.14 ± 0.81 ($p < 0.001$)	$\mathbf{94.20} \pm 1.17$ ($p < 0.001$)
Pen Digit	95.23 ± 1.37	94.67 ± 1.49	93.44 ± 1.77	95.57 ± 1.05 ($p = 0.006$)	95.75 ± 1.31 ($p < 0.001$)	$\mathbf{96.46}^{*} \pm 0.99$ ($p < 0.001$)
Satellite	93.28 ± 0.59	93.20 ± 0.69	91.38 ± 1.54	93.45 ± 0.63 ($p = 0.052$)	93.68 ± 0.70 ($p < 0.001$)	$\mathbf{94.07}^{*} \pm 0.61$ ($p < 0.001$)
Segment	92.26 ± 1.63	91.02 ± 1.92	93.89 ± 1.63	93.36 ± 1.52 ($p < 0.001$)	93.02 ± 1.44 ($p < 0.001$)	$\mathbf{94.66}^{*} \pm 1.55$ ($p < 0.001$)
Splice	78.91 ± 1.80	78.29 ± 2.50	$\mathbf{87.99} \pm 2.34$	82.66 ± 1.63 ($p < 0.001$)	79.63 ± 1.90 ($p < 0.001$)	$84.08^{*} \pm 1.71$ ($p < 0.001$)
Statlog	98.54 ± 0.88	98.19 ± 0.78	95.28 ± 15.08	98.85 ± 0.85 ($p = 0.004$)	99.00 ± 0.69 ($p < 0.001$)	$\mathbf{99.07} \pm 0.97$ ($p < 0.001$)
Svmguide1	95.29 ± 0.83	92.90 ± 4.30	94.59 ± 2.17	95.36 ± 0.80 ($p = 0.205$)	95.75 ± 0.75 ($p < 0.001$)	$\mathbf{96.30}^{*} \pm 0.52$ ($p < 0.001$)
Wine Quality	79.16 ± 0.82	79.40 ± 1.06	79.12 ± 0.96	79.43 ± 0.70 ($p = 0.190$)	80.01 ± 0.75 ($p < 0.001$)	$\mathbf{80.60}^{*} \pm 0.82$ ($p < 0.001$)

Table 7.10, the results of FS-KC are significantly better than those of KC for 5 datasets out of 10.

Table 7.10: Prediction accuracy for multi-class classification

Dataset	SVM	MKL 1	MKL 2	FS	KC	FS-KC
Crowd-sourced	89.75 ± 0.84	87.97 ± 1.28	87.28 ± 1.06	89.26 ± 0.91 ($p < 0.001$)	90.03 ± 0.89 ($p < 0.001$)	**90.35** ± 0.72 ($p < 0.001$)
Drive	81.01 ± 1.50	78.56 ± 2.07	31.31 ± 29.91	89.23 ± 5.85 ($p < 0.001$)	82.69 ± 1.43 ($p < 0.001$)	**92.34*** ± 2.72 ($p < 0.001$)
Letter	78.16 ± 1.15	67.70 ± 1.29	43.59 ± 8.77	79.96 ± 1.29 ($p < 0.001$)	79.73 ± 1.04 ($p < 0.001$)	**83.31*** ± 1.32 ($p < 0.001$)
Opt Digit	91.98 ± 1.09	92.34 ± 1.04	84.55 ± 3.34	90.46 ± 2.09 ($p < 0.001$)	92.92 ± 1.13 ($p < 0.001$)	**93.31*** ± 1.04 ($p < 0.001$)
Page Blocks	93.67 ± 2.86	93.65 ± 0.61	86.38 ± 17.10	93.75 ± 3.03 ($p = 0.892$)	94.61 ± 1.70 ($p < 0.001$)	**95.04** ± 2.20 ($p < 0.001$)
Pen Digit	90.40 ± 1.54	90.27 ± 1.35	88.99 ± 1.94	90.09 ± 1.75 ($p = 0.029$)	91.60 ± 1.46 ($p < 0.001$)	**91.82** ± 1.35 ($p < 0.001$)
Satellite	86.29 ± 0.66	86.44 ± 0.91	83.41 ± 2.19	86.07 ± 0.71 ($p = 0.016$)	**87.00** ± 0.76 ($p < 0.001$)	86.51 ± 1.61 ($p < 0.001$)
Segment	93.38 ± 1.24	93.25 ± 1.18	86.31 ± 18.62	93.43 ± 1.91 ($p = 0.874$)	93.94 ± 1.60 ($p < 0.001$)	**94.92*** ± 0.98 ($p < 0.001$)
Statlog	99.09 ± 0.73	98.60 ± 0.83	68.26 ± 27.20	99.14 ± 0.86 ($p = 0.764$)	99.22 ± 0.59 ($p < 0.001$)	**99.46*** ± 0.50 ($p < 0.001$)
Wine Quality	52.60 ± 4.07	56.50 ± 1.79	53.09 ± 3.44	52.42 ± 4.53 ($p = 0.792$)	57.54 ± 1.34 ($p < 0.001$)	**57.83** ± 1.57 ($p < 0.001$)

Analysis of an example GP individual An example program (individual) extracted from one of the GP runs is `(Exp (* K4 (+ K3 K1)(+ K1 K2)))` which is a symbolic expression describing the abstract syntax tree of an evolved kernel. When evaluated, the leaf nodes (e.g. `K1`) become kernel matrices. These are square symmetric matrices which in a way indicate the degree of similarity between each pair of the instances in the training data according to the corresponding kernel. Each kernel is measuring the similarity from a different perspective (note that kernels can have different subsets of features). The GP program is, in effect, combining all these various perspectives in order to obtain a matrix (at the root of the tree) that gives the degree of similarity between all pairs of instances. It is this new measure of similarity (evolved kernel) that contributes to the high performance of our proposed method.

Automatic construction of kernels involves a representation for a class kernel functions and a search mechanism to find a suitable kernel in the space. Considering that natural representations for a space of functions are syntax trees and that this space does not have properties that can be exploited by smarter search algorithms (such as

gradient descent), Genetic Programming remains the most natural tool to search for these functions. In Section 8.2, we also consider the extension of GP systems using MKL methods directly and an alternative approach of optimal kernel construction using neural networks.

7.5 Summary

We proposed a system for finding optimal kernels. The system has two subsystems: group feature selection and kernel construction.

For group feature selection, we conducted a experiment to compare the performance of GA with binary PSO and repeated the same experiments with the KNN classifier. We also conducted a experiment to compare the performance of sequential selection methods (SFS, SBS) with binary PSO. We proposed two methods: one taking an embedded approach based on the coefficients of the kernels in WR-MKL and the other taking a wrapper approach with the direct evaluation of fitness functions in a binary PSO method proposed in Chapter 4. The results show that the binary PSO approach is more flexible and accurate.

Then we conducted the experiments of optimal kernel construction. Genetic programming provided a way to tackle this difficult task. The results show that the proposed system achieves a significant improvement in comparison to the SVM with fine-tuned parameters.

Contributions and achievements:

- We proposed a new kernel construction system that optimizes subsets of feature and construction of kernels.
- We use WR-MKL (proposed in Chapter 6) to evaluate multiple kernels at once (based on their coefficients); that is, one run of SVM can reveal the relative importance of multiple kernels at once.
- Basic kernels (those kernel matrices that appear in leaf nodes) are also evaluated with WR-MKL before the GP run. Their relative importance is used to form a prior on the probability of their selection during the evolutionary process.
- We experimentally showed that the proposed method outperforms both SVM with fine-tuned hyper-parameters, and SVM with the previous kernel construction algorithms which do not include group feature selection process.

Part IV

Conclusions

Chapter 8

Conclusions

Our two main research goals were set as follows:

- to find new approaches to improve hyper-parameter optimization in SVMs;
- to find new approaches to improve optimal kernel construction in SVMs.

The new approaches are based on both methods in machine learning (ML) and methods in evolutionary computation (EC).

The main contributions of this research were arranged in two parts: hyper-parameter search in SVMs in Part II, and optimal kernel construction in SVMs in Part III.

In Part II, in Chapter 3, we developed a surface estimation method using the Bézier Curve methods, which improved the efficiency of a grid search. In Chapter 5, we proposed a discrete PSO with an adaptive calibration of evaluation points. Experiments showed that it further improved the efficiency without sacrificing accuracy.

In Part III, in Chapter 6, we introduced a new Multiple Kernel Learning method, called WR-MKL. Minimizing a ratio of margin and radius can be formulated as a standard one-level optimization problem. It is not convex with respect to the union of all parameters, but it is convex for each parameter. We proved that the optimization problem has a global minimum. We have also shown a method which derives a dual feasible solution and used it as a stopping criteria. The performance of the new MKL is comparable to the best ℓ_p-norm method having a fine-tuned parameter p, and it also has favorable properties such as producing sparse solutions and not needing to adjust additional parameters. In Chapter 7, we proposed a system to explore the space of possible nonlinear combinations of kernels and subsets of features to improve the prediction accuracy of SVMs. The experiments showed that the proposed method

achieves a significant improvement in comparison to the SVM having fine-tuned parameters.

Thus, we have successfully attained our research goals.

8.1 Chapter-wise Conclusions

In this section, for each of the two main parts, we provide more detailed chapter-wise conclusions.

8.1.1 Hyper-parameter Search using Bézier Curve Methods

We developed a surface estimation method using the Bézier Curve methods. We constructed cubic Hermite tensor product surfaces based on samples on a regular grid. An α-percent region was introduced to identify the most important search region, following which a finer second search was conducted on this α-percent region. The experiments showed that the optimal solutions of the proposed surface estimation methods are in most cases close to the optimal solutions obtained by exhaustively searching the space. That is, most often the proposed method can find optimal or near-optimal solutions for the hyper-parameters.

8.1.2 Discretization of Continuous PSO

We proposed a discretization of the continuous PSO in a unified way. We introduced discretization functions to combine the discrete version of PSO with the continuous PSO. We presented convergence theorems in the discrete spaces and derived some discrete (binary) PSO models from existing continuous PSO.

In the experiments of the binary PSO the proposed "position as probability" approach outperformed the "velocity as probability" approach. Therefore, our conclusion is that there is no reason not to use the "position as probability" approach which is theoretically supported. In the experiments of the discrete PSO, we showed the possibility of efficacy improvement without degrading the quality of the solutions. The heuristic methods for binary PSO and for discrete PSO improved the quality of the solutions in most cases.

Using this unified approach, the discrete (binary) PSO can benefit from various inventions and theoretical achievements in continuous PSO.

8.1.3 Hyper-parameter Search using Discrete PSO

We applied the discrete PSO model in Chapter 4 to the hyper-parameter search in SVMs. In order to further increase the efficiency, we introduced an adaptive calibration scheme of evaluation points. In the experiments, we showed that the standard PSO achieves excellent prediction accuracy for the hyper-parameter search in SVMs, and PSO with cellular fitness approximation can improve the efficiency without negatively impacting the accuracy. PSO also provides a means to automatically adjust the boundary of search region. PSO with cellular fitness approximation is more efficient and automatic than the surface estimation methods when the dimension of the search space is greater than two.

8.1.4 Weight-Radius Multiple Kernel Learning

We proposed a new MKL, called WR-MKL, which directly minimizes the ratio of the radius and the margin— the key terms in the generalization error rates of SVMs. It was formulated as the combination of SVM and SVDD. We proved that it is a closed system and has a global optimal solution. Experiments showed that the performance of the proposed method is comparable to the ℓ_p-norm method having the fine-tuned parameter p. The proposed WR-MKL has two additional favorable properties: it produces sparse solutions; and it does not need to adjust additional parameters.

8.1.5 Optimal Kernel Construction

We presented a system to explore the large space of nonlinear combinations of kernels and subsets of features. We conducted preliminary experiments of feature subset selection. Firstly we compared the performance of two binary PSO models (B-SPSO and B-FLRPSO) introduced in Chapter 4 with that of GA. We also repeated the experiment using k nearest neighbor instead of SVM. Then we carried out an another experiment to compare a heuristic feature subset selection method based on MKL with B-SPSO which recorded the best performance in the previous experiments. The former represents the embedding approach and the latter represents the wrapper approach. As a result, we verified that the wrapper approach which uses the direct measurements is more accurate than the embedding approach which uses the weights of kernels as the evaluation of the relevance of features. According to those preliminary experiments, binary PSO is accurate and flexible for feature subset selection. Also GP is an effective tool for constructing mathematical expressions that define new kernel functions. Our

results showed that the proposed system, with two subsystems for selecting subsets of features and constructing kernels, achieves a significant improvement in comparison to the SVM having fine-tuned parameters.

8.2 Future Works

This section outlines some possible research directions for the proposed methods in Chapters 3, 5, 4, 6 and 7.

8.2.1 Hyper-parameter Search using Bézier Curve Methods and Discrete PSO

One possible future direction is to combine the surface estimation method proposed in Chapter 3 with the adaptive discrete PSO introduced in Chapter 5. The idea is to construct a prediction surface and use the information to direct the search (to decide the most relevant region for the finer search). It is possible to efficiently construct the surface. Since the calibration between evaluation points are changed along each axis, the evaluation points keep the regular pattern even after the dynamic changes of calibration. The challenge is that at the beginning of the search we have to rely on a rougher estimation and all of this must be done with efficiency in mind.

8.2.2 Hyper-parameter Search using PSO versus Bayesian Optimization Methods

In Chapter 5, we showed that the overall performance of PSO is excellent for the task of hyper-parameter search. Computational complexity of PSO increases linearly with respect to the increase of dimensionality of spaces. We expect that PSO can be applicable in higher dimensional spaces.

The surface estimation methods using Bézier curve methods in Chapter 3 can be applicable up to five dimensional spaces. Beyond that dimension we need to rely on other baseline methods. The first choice would be Bayesian optimization methods in Section 2.10.1. Through the experiments for the comparison between the PSO methods and the Bayesian optimization methods we would like to examine the better search methods in high dimensional spaces.

8.2.3 Weight-Radius Multiple Kernel Learning

New WR-MKL Model

In Appendix A.1, we show that our approach of WK-MKL models brings some important advantages in the light of Theorem 2.2. When applying Theorem 2.2 to our models, the hyper-parameter λ and the regularization error rates are estimated based on the ratio of margin and radius and the training size n. Using MKL, the best estimation of kernel hyper-parameters can be automatically constructed based on the the candidate sets of kernel hyper-parameters. Therefore our approach will lead to convenient MKL models in which optimal specification of all hyper-parameters are estimated automatically.

The largest problem of our models is the usage of the block coordinate methods which may take a long time for convergence. In (Shalev-Shwartz and Ben-David [137]), the authors apply the SGD (Section 2.3.5) to minimize the risk of neural network (Chapter 20 in [137]). In the case of neural network the loss function is highly non-convex. However, they show that the application of SGD is a reasonable solution. In our case the optimization problem is not convex for the whole set of parameters but convex for each parameter. Using the SGD and this "partial convex property", we hope that the optimization problem can be solved in a single iteration.

Estimation of Better Parameter Sets in MKL

In the experiments in Chapter 6 and Chapter 7, we verify that in general the MKL methods do not necessary improve the accuracy of SVM with fine-tuned parameters. A question is how we should construct a set of kernel hyper-parameters to improve the prediction accuracy of MKL. To address this question we consider a system of multiple GPs which is described below.

8.2.4 Optimal Kernel Construction

System of Multiple GPs

In the experiments in Chapter 7, we used the top $n_{FS} = 30$ subsets of features and the top $n_{KC} = 30$ nonlinear kernels. Scaling up the values of n_{FS} and n_{KC} is a valuable direction for future research. Here we consider a system which consists of multiple GPs. The purpose of the system is to scale up the experiments of kernel construction and also address the improvement of prediction accuracy of MKL. In the experiments

in Chapter 7, the final output of GP was a best estimation of optimal kernel. However using this system we can output both the best estimation of the optimal kernel and the optimal set of kernels.

The system consists of a set of GP subsystems. In each GP subsystem we run MKL once to evaluate the fitness of the kernels (based on their coefficients). Therefore, if there are m GP subsystems, we obtain all fitness evaluations in the system by running MKL m times.

We can introduce a topology in the whole system such as the ring topology or the gbest topology (Chapter 4). Each GP subsystem has its neighbor and exchanges the information with the members in the neighbor. For the crossover operation we can consider two types of crossover; the crossover within each GP and the crossover between its own GP and the neighbor GP. For the latter type of crossover we can set a rule such that one parent must be chosen from its own GP.

The final outputs of the system are the best estimation of optimal kernel in the whole system and a set of optimal kernels from the best GP subsystem. This design can scale up the number of individuals in the whole system and also address the question about how to improve the prediction accuracy of MKL.

Function Approximation by Neural Networks

Neural networks are universal approximators. That is, let $f : [-1,1]^n \Rightarrow [-1,1]$ be a Lipschitz function. For some $\epsilon > 0$, we can construct a neural network $N : [-1,1]^n \Rightarrow [-1,1]$ such that for every $\mathbf{x} \in [-1,1]^n$, $|f(\mathbf{x}) - N(\mathbf{x})| \leq \epsilon$ (Exercise 20.1 in Shalev-Shwartz and Ben-David [137]). However, the minimum size (the number of nodes) in the network is exponential in n (Theorem 20.5 in [137]). Furthermore, if the number of nodes in the hidden layer is greater than 3, it is NP hard to implement ERM (Empirical Risk Minimization) rule in the neural network (Theorem 20.7 in [137]).

In practice, neural networks have been used as state-of-art systems in various applications such as image processing and speech recognition. For the task of function approximation, Liang and Srikant [89] show that the number of nodes needed in a shallow network whose depth does not depend on the approximation error ϵ is exponentially larger than that in a deep network whose depth grows with $\frac{1}{\epsilon}$.

In Section 6.2.2, shift-invariant kernels ($k(\mathbf{x},\mathbf{y}) = \kappa(\mathbf{x}-\mathbf{y})$ in Section 2.3.3) have the

following integral representation:

$$\begin{aligned}\kappa(\mathbf{x}-\mathbf{y}) &= \int_{\mathbb{R}^d} p(\boldsymbol{\omega}) e^{i\boldsymbol{\omega}^T(\mathbf{x}-\mathbf{y})} d\boldsymbol{\omega} \\ &= \mathbb{E}_{\boldsymbol{\omega}}\left[e^{i\boldsymbol{\omega}^T\mathbf{x}} e^{-i\boldsymbol{\omega}^T\mathbf{y}}\right].\end{aligned}$$

In more general sense, the following theorem holds.

Theorem 8.1. *(Dai et al. [31], Hein and Bousquet [63])*
If $k(\mathbf{x},\mathbf{y})$ is a PDS function (a kernel), then there exists a set Ω, a measure p on Ω, and a random function $\phi_\omega(\mathbf{x})$ from $L_2(\Omega,p)$ such that $k(\mathbf{x},\mathbf{y}) = \int_\Omega \phi_\omega(\mathbf{x})\phi_\omega(\mathbf{y})dp(\omega)$.

Conversely,

Theorem 8.2. *(Dai et al. [31], Wendland [166])*
If $k(\mathbf{x},\mathbf{y}) = \int_\Omega \phi_\omega(\mathbf{x})\phi_\omega(\mathbf{y})dp(\omega)$ for a nonnegative measure $p(\omega)$ on Ω, and $\phi_\omega(\mathbf{x}) \in L_2(\Omega,p)$, then $k(\mathbf{x},\mathbf{y})$ is a PDS function (a kernel).

Therefore, exploring the function space in $L_2(\Omega,p)$ is equivalent to exploring the space of kernels. Using SGD methods in Section 2.3.5, we can solve nonlinear SVMs using the function $\phi_\omega(\mathbf{x})$ instead of the kernel matrix. This approach solves the scale problem of kernel matrix (Dai et al. [31]). Thus, if we can explore the function space in $L_2(\Omega,p)$ using neural networks efficiently, we can explore the space of kernels efficiently, which provides an alternative approach for the optimal kernel construction.

8.2.5 Discrete PSO for Problems of Optimal Permutations

In Appendix A.2, we show that a sequence of integers without repetition can be expressed using matrix whose entries are 0 or 1. Here we consider the case of vertex-base representation. The matrix has only one 1 in each row and each column. The difference $\mathbf{x}_2 - \mathbf{x}_1$ of a sequence $\mathbf{x}_2$ and a sequence $\mathbf{x}_1$ represents an operation which transforms the sequence $\mathbf{x}_1$ to the sequence $\mathbf{x}_2$. In this way we represent the position (sequence) and the velocity (difference) using matrix.

The velocity can be decomposed a union of circular permutations. If the velocity $\mathbf{x}_2 - \mathbf{x}_1$ is decomposed as a union of m circular permutations, then we have a $2^m - 1$ intermediate positions between the position $\mathbf{x}_1$ to the position $\mathbf{x}_2$. In the standard operations of PSO we explore the local space represented by a linear combination of the current velocity, $(pbest - \mathbf{x}_1)$ and $(gbest - \mathbf{x}_1)$ (Figure 2.6). In the matrix case how should we explore the local space around the current position? For example, we can

explore the space consisting of the following components in the current iteration and in the previous iteration:

- $(pbest - \mathbf{x}_1)$: the intermediate positions between the $\mathbf{x}_1$ to the position $pbest$
- $(gbest - \mathbf{x}_1)$: the intermediate positions between the $\mathbf{x}_1$ to the position $gbest$.

To tackle the combinatorial optimization problems using PSO the matrices are promising tools to represent the problems and efficiently solve them. Since matrices are looked upon as an extension of the real number system in linear algebra, it would bring a lot of possibilities to solve the combinatorial optimization problems. To approach the better formulations and the better understanding, we would have to work hard to examine various ideas. We hope that the good balance of PSO between the exploration and the exploitation is also effective in the matrix spaces. In other words, simple algorithms would work well to achieve good results in the optimization problems.

Appendix A

Supplements

In Appendix A.1 we present a new WR-MKL model, and in Appendix A.2 we present a new discrete PSO model for "Travaling Salesman Problem" (TSP).

A.1 New WR-MKL Model

In this section we discuss a model which we are currently working on. The model in (A.1) below is slightly different from the model (6.24). We highlight the differences from the model 6.24 with the blue colour.

$$\begin{aligned}
&\min_{\boldsymbol{\theta},\mathbf{w},b,R,\mathbf{a},\boldsymbol{\xi}} \frac{\lambda}{2}\sum_{k=1}^{m}\|\mathbf{w}_k\|^2 R + \frac{1}{n}\sum_{i=1}^{n}\xi_i \\
&\quad \text{s.t. } y_i(\sum_{k=1}^{m}\langle \mathbf{w}_k, \theta_k\Phi_k(\mathbf{x}_i) - \mathbf{a}_k\rangle + b) \geq 1-\xi_i, \qquad i=1,\ldots,n \\
&\quad\quad \xi_i \geq 0, \quad i=1,\ldots,n. \\
&\quad\quad \sum_{k=1}^{m}\|\theta_k\Phi_k(\mathbf{x}_i) - \mathbf{a}_k\|^2 \leq R,\ i=1,\ldots,n
\end{aligned} \tag{A.1}$$

It moves the origin of the space to the centre of the sphere and added $\frac{1}{n}$ to the model to conform to the settings in Theorem 2.2. It is equivalently formulated in the unconstrained form:

$$\begin{aligned}
&\min_{\mathbf{w},\boldsymbol{\theta}} \frac{\lambda}{2}\sum_{k=1}^{m}\|\mathbf{w}_k\|^2 \max_i \sum_{k=1}^{m}\|\theta_k\Phi_k(\mathbf{x}_i) - \mathbf{a}_k\|^2 \\
&\quad + \frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\sum_{k=1}^{m}\langle \mathbf{w}_k, \theta_k\Phi_k(\mathbf{x}_i) - \mathbf{a}_k\rangle + b)\right)
\end{aligned} \tag{A.2}$$

For multi-class classification problem, we prepare a kernel for each class:

$$\mathbf{w}_k = \left[\mathbf{w}_{k,1}, \ldots, \mathbf{w}_{k,h}\right].$$

The optimization problem is formulated as:

$$\begin{aligned}\min_{\mathbf{w},\boldsymbol{\theta}} \frac{\lambda}{2} \sum_{j=1}^{h} \left(\sum_{k=1}^{m} \|\mathbf{w}_{k,j}\|^2 \max_i \sum_{k=1}^{m} \|\theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k\|^2 \right) \\ + \frac{1}{n} \sum_{i=1}^{n} \max \left(0, 1 + \max_{j \neq y_i} \sum_{k=1}^{m} \langle \mathbf{w}_{k,j}, \theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k \rangle - \sum_{k=1}^{m} \langle \mathbf{w}_{k,y_i}, \theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k \rangle \right)\end{aligned} \tag{A.3}$$

We will show that in light of Theorem 2.2 our models bring some important advantages. For convenience we rewrite the theorem.

Theorem A.1. *Corollary 13.9 in [137]*
Let the loss function be ρ-Lipschitz and $\mathbf{w}$ is bounded and $\|\mathbf{w}\|^2 \leq B^2$. For any training set size n, let $\lambda = \sqrt{\frac{4\rho^2}{B^2 n}}$. Then the RLM rule with the regularization function $\frac{\lambda}{2}\|\mathbf{w}\|^2$ satisfies

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right] \leq \min_{\mathbf{w}} L_{\mathcal{D}}(\mathbf{w}) + \frac{3\rho B}{\sqrt{n}}.$$

In particular, for every $\epsilon > 0$, if $n \geq \frac{9\rho^2 B^2}{\epsilon^2}$ then for every distribution $\mathcal{D}$, $\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right] \leq \min_{\mathbf{w}} L_{\mathcal{D}}(\mathbf{w}) + \epsilon$.

The theorem tells us that if we know ρ and B, we can estimate the optimal specification of the hyper-parameter λ. However, in the case of nonlinear SVM, $\rho^2 = \|\Phi(\mathbf{x}_i)\|^2$. It may not be easy to bound the quantity $\|\Phi(\mathbf{x}_i)\|^2$ by an appropriate value B. We have the same problem for Theorem 2.7 in Section 2.3.5.

Now we apply this theorem to our model (A.2). Let

$$\begin{aligned} g(\mathbf{w}, \boldsymbol{\theta}, \mathbf{a}) &= \frac{\lambda}{2} \sum_{k=1}^{m} \|\mathbf{w}_k\|^2 \max_i \sum_{k=1}^{m} \|\theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k\|^2 \\ g_1(\mathbf{w}) &= \sum_{k=1}^{m} \|\mathbf{w}_k\|^2 \\ g_2(\boldsymbol{\theta}, \mathbf{a}) &= \max_i \sum_{k=1}^{m} \|\theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k\|^2 \end{aligned}$$

$g(\mathbf{w}, \boldsymbol{\theta}, \mathbf{a})$ is $\lambda g_2(\boldsymbol{\theta}, \mathbf{a})$-strongly convex with respect to $\mathbf{w}$. In the loss function;

$$\sum_{k=1}^{m} \langle \mathbf{w}_k, \theta_k \Phi_k(\mathbf{x}_i) - \mathbf{a}_k \rangle$$

is ρ_1-Lipschitz with respect to $\mathbf{w}$ where $\rho_1 = \|(\theta_1\Phi_1(\mathbf{x}_i) - \mathbf{a}_1, \ldots, \theta_m\Phi_m(\mathbf{x}_i) - \mathbf{a}_m)\|$ by Cauchy-Shwarz inequality. Since

$$\|(\theta_1\Phi_1(\mathbf{x}_i) - \mathbf{a}_1, \ldots, \theta_m\Phi_m(\mathbf{x}_i) - \mathbf{a}_m)\| \leq \max_i \|(\theta_1\Phi_1(\mathbf{x}_i) - \mathbf{a}_1, \ldots, \theta_m\Phi_m(\mathbf{x}_i) - \mathbf{a}_m)\|,$$

the loss function is ρ_2-Lipschitz with respect to $\mathbf{w}$ and $\rho_2^2 = g_2(\boldsymbol{\theta}, \mathbf{a})$. In our model, the inequality (2.5) is simplified to

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right] \leq L_{\mathcal{D}}(\mathbf{w}^*) + \frac{\lambda}{2} g_1(\mathbf{w}) g_2(\boldsymbol{\theta}, \mathbf{a}) + \frac{4}{\lambda n} \tag{A.4}$$

Suppose that $\|g_1(\mathbf{w})g_2(\boldsymbol{\theta}, \mathbf{a})\| \leq B_g^2$. Substituting $\lambda = \sqrt{\frac{4}{B_g^2 n}}$ into (A.4),

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)})\right] \leq \min_{\mathbf{w}} L_{\mathcal{D}}(\mathbf{w}) + \frac{3B_g}{\sqrt{n}}.$$

The same relation holds for $\mathbf{a}$,

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{a}_{A(S)})\right] \leq \min_{\mathbf{a}} L_{\mathcal{D}}(\mathbf{a}) + \frac{3B_g}{\sqrt{n}}.$$

To combine those results, let $(\mathbf{w}^*, \mathbf{a}^*) = \operatorname{argmin}_{(\mathbf{w},\mathbf{a})} L_{\mathcal{D}}((\mathbf{w}, \mathbf{a}))$. Since

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)}, \mathbf{a}^*)\right] \leq L_{\mathcal{D}}((\mathbf{w}^*, \mathbf{a}^*)) + \frac{3B_g}{\sqrt{n}}$$

and

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)}, \mathbf{a}_{A(S)})\right] \leq L_{\mathcal{D}}((\mathbf{w}_{A(S)}, \mathbf{a}^*)) + \frac{3B_g}{\sqrt{n}},$$

we derive

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)}, \mathbf{a}_{A(S)})\right] \leq \min_{(\mathbf{w},\mathbf{a})} L_{\mathcal{D}}((\mathbf{w}, \mathbf{a})) + \frac{6B_g}{\sqrt{n}}.$$

For $\boldsymbol{\theta}$, $\|\Phi_k(\mathbf{x}_i)\|^2 = K_k(i,i)$ takes different values for each k except for the case of Gaussian kernel. Therefore we need to evaluate each θ_k separately. As a result,

$$\mathbb{E}_{\mathcal{S}}\left[L_{\mathcal{D}}(\mathbf{w}_{A(S)}, \boldsymbol{\theta}_{A(S)}, \mathbf{a}_{A(S)})\right] \leq \min_{(\mathbf{w},\boldsymbol{\theta},\mathbf{a})} L_{\mathcal{D}}((\mathbf{w}, \boldsymbol{\theta}, \mathbf{a})) + \frac{(6+3m)B_g}{\sqrt{n}}.$$

where m is the number of kernels. We also obtain the same results for the multi-class classification model (A.3). The application of Theorem 2.7 into our models also reduces to a simple formulation.

In our model since B_g is the quantity we try to minimize, our models will lead to a better generalization error rate. Using our model λ is also simply specified as $\lambda = \sqrt{\frac{4}{B_g^2 n}}$. Therefore, our models (A.2) and (A.3) simplify the formulation in Theorem 2.2 and Theorem 2.7, give the better bound, and implement them without any restrictions.

A.2 New Approach for Traveling Salesman Problem

In Section 4.4.4, we have shown that the discrete PSO with round functions can be used to solve the problems of the optimal sequences of integers with repetitions. A question is whether it is possible to extend the discrete PSO model to solve the problems of the optimal permutations (the optimal sequences of integers without repetitions). It turns out, however, that it are a more difficult problem. In this section we show an idea to configure the discrete PSO for the problems of optimal permutations. We consider "Traveling Salesman Problem" (TSP).

A.2.1 Traveling Salesman Problem (TSP)

We start with a brief review of definitions and basic theorems of Traveling Salesman Problem (TSP). Using the terminology of graph theory, d cities are represented as vertices i for $i = 1, \ldots, d$ and a city i and a city j are connected by the edge (i, j) with the associated cost $|(i, j)|$. If $|(i, j)| = |(j, i)|$ for all $i, j = 1, \ldots, d$, the problem is said to be symmetric. In this section we consider only the symmetric TSP. A cycle is a set of edges $\{(i_1, i_2), \ldots, (i_k, i_1)\}$ with $i_p \neq i_q$ for all $p \neq q$. A tour is a cycle with $k = d$. For each tour T, the length $L(T)$ is the sum of the costs of the edges in T, $L(T) = \sum_{(i,j)\in T} |(i, j)|$. The traveling salesman problem is an optimization problem which seeks for an optimal tour with the minimum length.

Definition A.1. *k-exchange: A tour T' is a k-exchange of a tour T, if T' can be obtained by deleting k edges from T and reconnecting by new k edges.*

Definition A.2. *k-opt: A tour T is a k-opt if it is impossible to obtain a shorter tour T' which is a k-exchange of a tour T.*

Definition A.3. *k-opt algorithm: A k-opt algorithm is an algorithm in which a shorter tour is obtained in each step by applying the operation of k-exchange.*

We present two basic theorems in TSP.

Theorem A.2. *(Mak and Morton [102])*
Every tour T' which is a k-exchange of a tour T is obtained by applying k or fewer 2-exchanges to T.

For a tour T' which is a k-exchange of a tour T, let $X = \{x_1, \ldots, x_k\}$ be a set of k deleted edges and $Y = \{y_1, \ldots, y_k\}$ be a set of k added edges. Denote the cost of

x_i and y_i be $|x_i|$ and $|y_i|$ respectively, and define $g_i = |x_i| - |y_i|$. If $L(T') < L(T)$, $L(T) - L(T') = \sum g_i > 0$. In general, the following theorem holds.

Theorem A.3. *(Lin and Kernighan [91])*
If a sequence of numbers $(g_1, \ldots, g_k)$ has a positive sum $(g_1 + \cdots + g_k > 0)$, there is a cyclic permutation of these numbers such that every partial sum $g_1 + \cdots + g_s, \ \ s \leq k$ is positive.

To clarify the meaning of the theorem, we show the short proof in (Lin and Kernighan [91]).

Proof. Let l be the largest index such that $g_1 + \cdots + g_{l-1}$ is minimum.
If $l \leq j \leq k$,

$$g_l + \cdots + g_j = g_1 + \cdots + g_j - (g_1 + \cdots + g_{l-1}) > 0.$$

If $1 \leq j < l$,

$$g_l + \cdots + g_k + g_1 + \cdots + g_j \geq g_l + \cdots + g_k + g_1 + \cdots + g_{l-1} > 0.$$

□

A.2.2 Discrete PSO with Swap Operation

To deal with TSP, Clerc [26] proposed a discrete PSO based on an exchange of two vertices i and j, which is called a swap operation. (Using the terminology of TSP, a swap operation is a 4-exchange if the vertices are not adjacent each other, and a 2-exchange if the vertices are adjacent.) In this system a position $\mathbf{x}$ of a particle is defined as a sequence of vertices $\mathbf{x} = (i_1, \ldots, i_d)$ and velocity $\mathbf{v}$ is defined as a sequence of swap operations $\mathbf{v} = \{(i_1, j_1), \ldots, (i_k, j_k)\}$. The arithmetics of position and velocity are defined as follows (Clerc [26]):

- opposite (minus) of velocity:
  $-\mathbf{v} = \{(i_k, j_k), \ldots, (i_1, j_1)\}$
- position plus velocity: $\mathbf{x}' = \mathbf{x} + \mathbf{v}$ which applies the sequence of swap operations in the velocity to the position vector
- position minus position: $\mathbf{x}' - \mathbf{x} = \mathbf{v}$
- velocity plus velocity: concatenation of swap operations in each velocity
- constant times velocity:

- Case $c = 0$: $c\mathbf{v} = \emptyset$
- Case $0 < c \leq 1$: $c\mathbf{v} = \{(i_1, j_1), \ldots, (i_s, j_s)\}$
  where $s = \lfloor ck \rfloor$
- Case $c > 1$: let $b = \lfloor c \rfloor$ and $c = b + c'$.
  $c\mathbf{v} = \sum_{i=1}^{b} \mathbf{v} + c'\mathbf{v}$

Based on those rules, the velocity update rule of PSO is defined as

$$\mathbf{v}^{t+1} = \mathbf{v}^t + c\left(\hat{\mathbf{p}}^t - \mathbf{x}^t\right)$$

where

$$\hat{\mathbf{p}}^t = \mathbf{p}^t + \frac{1}{2}(\mathbf{g}^t - \mathbf{p}^t).$$

It is a quite interesting model. However, it is far more difficult to deal with the space of swap operations compared with the Euclidean space. Swap operations are not commutative and an expression of velocity is not unique. Furthermore, the computational complexity of the "position minus position" operation in dimension d is $o(d^2)$. Most articles, however, have adopted this discrete PSO model for TSP (Shi et al. [139], Goldbarg et al. [54], Akhand et al. [101]).

A.2.3 Discrete PSO with Rank Functions

An another approach for TSP (the problem of optimal permutations) is to use continuous PSO (Verma and kumar [163], Liu et al. [93], Dubey and Gupta [40]). In this approach we use the rank of a position vector $\mathbf{x} \in \mathbb{R}^d$ to represent a permutation of integers. For instance, a position vector $\mathbf{x} = (2.99, 1.81, 0.05, 3.72, 2.10, 0.68)$ is converted to the permutation $(5, 3, 1, 6, 4, 2)$. (Since $x_3 = 0.05$ is the smallest value, x_3 is picked first and assigned the rank 1. $x_6 = 0.68$ is the second smallest value, x_6 is picked next and assigned the rank 2, and so on.) This operation divides the space into $d! = d \times (d-1) \times \cdots \times 1$ regions each of which is represented as

$$\{\mathbf{x} \mid x_{i_1} \geq x_{i_2} \geq \cdots \geq x_{i_d}\} \ .$$

where $(i_1, \ldots, i_d)$ be the permutation of integers $(1, \ldots, d)$.

We can define another discretization function based on the rank function. Suppose that $\mathbf{x} \in [0, c]^d$ with positive integer $c > 0$. The idea of the discretization function is to map $\mathbf{x}$ into an evaluation point in the same domain (space). Therefore, we just scale the

rank of $\mathbf{x}$. Let $r(\mathbf{x})$ be the rank function which maps x_j to the rank of x_j in $(x_1, \ldots, x_d)$. We denote $r(\mathbf{x}) = (r_1(\mathbf{x}), \ldots, r_d(\mathbf{x}))$. Then we define a discretization function:

$$\psi(\mathbf{x}) = \left(\frac{c \times r_1(\mathbf{x})}{d}, \ldots, \frac{c \times r_d(\mathbf{x})}{d} \right). \tag{A.5}$$

For instance, for the above $\mathbf{x} = (2.99, 1.81, 0.05, 3.72, 2.10, 0.68)$, $c = 4$ and $d = 6$.

$$\psi(\mathbf{x}) = \left(\frac{5 \times 4}{6}, \frac{3 \times 4}{6}, \frac{1 \times 4}{6}, \frac{6 \times 4}{6}, \frac{4 \times 4}{6}, \frac{2 \times 4}{6} \right). \tag{A.6}$$

The rank function maps $\mathbf{x}$ into one of the $d!$ regions and the discretization function ψ maps $\mathbf{x}$ into an evaluation point in the region.

However, it turns out that this model is less accurate compared with the discrete PSO with swap operations. In TSP, the cost for swapping every pair of edges should be measured based only on the weights of the edges. However, in our model the swapping of the pair of elements $(\frac{1\times4}{6}, \frac{6\times4}{6})$ is less likely to occur than the swapping of $(\frac{3\times4}{6}, \frac{4\times4}{6})$ in (A.6), because particles need to travel more distance for the former to occur. In other words, we are imposing the Euclidean metric which is unnecessary in TSP.

A.2.4 Discrete PSO with Matrix Representations

To overcome this problem we examine the matrix representation of TSP.

Vertex-base Representation: Firstly we consider the vertex-base representation of permutations. In Table A.1, each row represents a vertex and each column represents a position of a sequence (permutation). Table A.1 represents a permutation $p1 = (5, 6, 1, 4, 3, 2)$, in which the vertex 5 occupies the first position of the permutation and vertex 6 occupies the second position of the permutation. The entries in the table have values 0 or 1 and only one entry has the value 1 in each row and in each column.

Let's see an another permutation $p2 = (1, 6, 5, 4, 3, 2)$ which is obtained by swapping $v1$ and $v5$ in Table A.2. The velocity (difference) $p2 - p1$ is computed as in Table A.3, which represents a swap operation.

Table A.4 is an another permutation $p3 = (2, 1, 6, 3, 4, 5)$. The velocity $p3 - p1$ is computed as in Table A.5, which is decomposed as a union of three swap operations ($v1$,$v6$), ($v3$,$v4$) and ($v2$,$v5$).

Table A.1: Vertex-base Representation of Permutation $p1$

	p1	p2	p3	p4	p5	p6
v1	0	0	1	0	0	0
v2	0	0	0	0	0	1
v3	0	0	0	0	1	0
v4	0	0	0	1	0	0
v5	1	0	0	0	0	0
v6	0	1	0	0	0	0

Table A.2: Vertex-base Representation of Permutation $p2$

	p1	p2	p3	p4	p5	p6
v1	1	0	0	0	0	0
v2	0	0	0	0	0	1
v3	0	0	0	0	1	0
v4	0	0	0	1	0	0
v5	0	0	1	0	0	0
v6	0	1	0	0	0	0

Table A.3: Velocity $p2 - p1$

	p1	p2	p3	p4	p5	p6
v1	1	0	-1	0	0	0
v2	0	0	0	0	0	0
v3	0	0	0	0	0	0
v4	0	0	0	0	0	0
v5	-1	0	1	0	0	0
v6	0	0	0	0	0	0

Table A.6 shows yet an another permutation $p4 = (1, 2, 3, 4, 6, 5)$. The velocity $p4 - p3$ is computed as in Table A.7, which is decomposed as a union of a swap operation ($v1$,$v2$) and a cyclic permutation of ($v3$,$v4$,$v6$).

In general, the velocities are decomposed as a union of cyclic permutations. Each cyclic permutation can be computed as in Algorithm A.1.

In this model, positions of particles (permutations) are represented as matrices. The velocity is computed as the difference of the matrices and decomposed as a union of cyclic permutations. Based on these components we can construct a discrete PSO

Table A.4: Vertex-base Representation of Permutation $p3$

	p1	p2	p3	p4	p5	p6
v1	0	1	0	0	0	0
v2	1	0	0	0	0	0
v3	0	0	0	1	0	0
v4	0	0	0	0	1	0
v5	0	0	0	0	0	1
v6	0	0	1	0	0	0

Table A.5: Velocity $p3 - p1$

	p1	p2	p3	p4	p5	p6
v1	0	1	-1	0	0	0
v2	1	0	0	0	0	-1
v3	0	0	0	1	-1	0
v4	0	0	0	-1	1	0
v5	-1	0	0	0	0	1
v6	0	-1	1	0	0	0

Table A.6: Vertex-base Representation of Permutation $p4$

	p1	p2	p3	p4	p5	p6
v1	1	0	0	0	0	0
v2	0	1	0	0	0	0
v3	0	0	1	0	0	0
v4	0	0	0	1	0	0
v5	0	0	0	0	0	1
v6	0	0	0	0	1	0

model for TSP which is more efficient than the discrete PSO with swap operations.

Edge-base Representation: Next we consider the edge-base representation of permutations. It is a little more complicated than the vertex-base representation, but the edge-base representation is a common approach in TSP (Section A.2). In this case the minimal operation is 2-exchange in Figure A.1.

Edges among cities are represented by incident matrices. In Table A.8, each row represents each city whose edges are connected to two cities and the columns with an

Algorithm A.1: Find a Cyclic Permutation in Velocity

Require: Velocity

Start at any row r_0 with nonzero elements.

Let $PERM = \{r_0\}$

Let c_0 be the column of the element 1. Let c_{-1} be the column of the element -1.

while $true$ **do**

In the column c_{-1} find a row r_1 which has an element 1.

$PERM = PERM \bigcup \{r_1\}$

In the row r_1, find a column c_{-1} which has an element -1.

if $c_{-1} = c_0$ **then**

Break;

end if

end while

return $PERM$

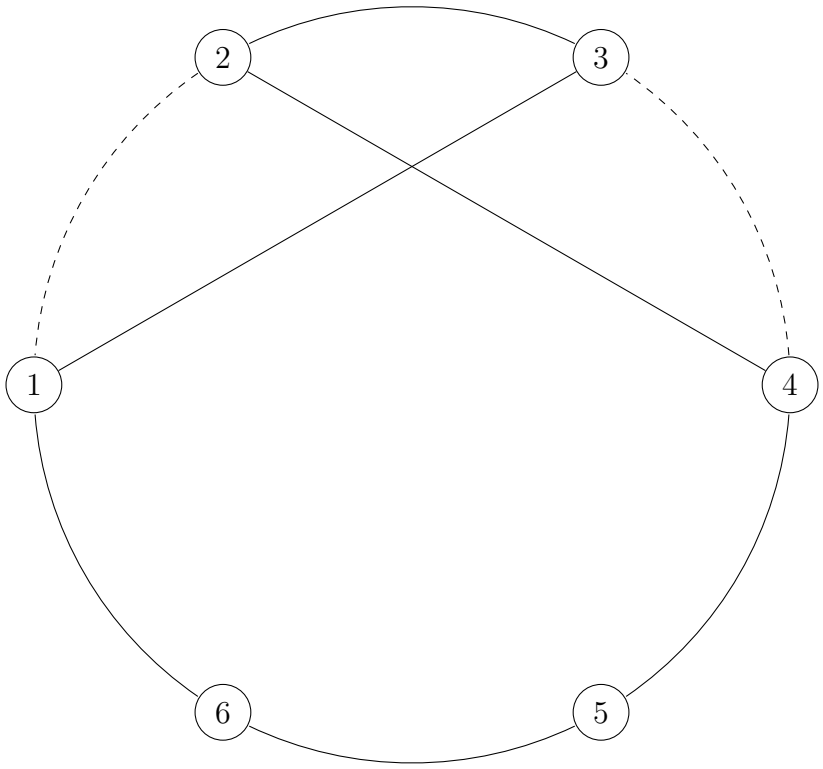


Figure A.1: 2-exchange in TSP ($per2$)

Table A.7: Velocity $p4 - p3$

	p1	p2	p3	p4	p5	p6
v1	1	-1	0	0	0	0
v2	-1	1	0	0	0	0
v3	0	0	1	-1	0	0
v4	0	0	0	1	-1	0
v5	0	0	0	0	0	0
v6	0	0	-1	0	1	0

element 1 represent the cities connected by the edges. Table A.8 represents a permutation $per1 = (1, 2, 3, 4, 5, 6)$ and Table A.9 represents the 2-exchange depicted in Figure A.1. The tables are symmetric and have two 1s in each row and in each column.

Table A.8: Edge-base Representation of Permutation $per1$

	v1	v2	v3	v4	v5	v6
v1	0	1	0	0	0	1
v2	1	0	1	0	0	0
v3	0	1	0	1	0	0
v4	0	0	1	0	1	0
v5	0	0	0	1	0	1
v6	1	0	0	0	1	0

Table A.9: Edge-base Representation of Permutation $per2$

	v1	v2	v3	v4	v5	v6
v1	0	0	1	0	0	1
v2	0	0	1	1	0	0
v3	1	1	0	0	0	0
v4	0	1	0	0	1	0
v5	0	0	0	1	0	1
v6	1	0	0	0	1	0

The velocity $per2 - per1$ is computed as in Table A.10, which represents a minimal 2-exchange operation. It is also symmetric and each row and each column sums to zero.

Table A.11 represents a 3-exchange in Figure A.2. The velocity $per3 - per1$ is computed in Table A.12. Algorithm A.2 shows how to identify a k-exchange by traversing

Table A.10: Velocity $per2 - per1$

	v1	v2	v3	v4	v5	v6
v1	0	-1	1	0	0	0
v2	-1	0	0	1	0	0
v3	1	0	0	-1	0	0
v4	0	1	-1	0	0	0
v5	0	0	0	0	0	0
v6	0	0	0	0	0	0

Table A.11: Edge-base Representation of Permutation $per3$

	v1	v2	v3	v4	v5	v6
v1	0	0	0	1	0	1
v2	0	0	1	0	1	0
v3	0	1	0	0	0	1
v4	1	0	0	0	1	0
v5	0	1	0	1	0	0
v6	1	0	1	0	0	0

Table A.12: Velocity $per3 - per1$

	v1	v2	v3	v4	v5	v6
v1	0	-1	0	1	0	0
v2	-1	0	0	0	1	0
v3	0	0	0	-1	0	1
v4	1	0	-1	0	0	0
v5	0	1	0	0	0	-1
v6	0	0	1	0	-1	0

elements in a velocity (a difference between two permutations). In the algorithm, X is a set of k deleted edges and Y is a set of k added edges. We start with a city A. At the city A we delete an edge and add an edge which connects to a city B. We move to the city B. At the city B we delete an edge and add an edge which connects to a city C. We move to the city C. At the city C we delete an edge. If the deleted edge at C is the same as the deleted edge at A, we have return to the city A and obtained a 2-exchange. Otherwise, we contnue the process until we return to the city A.

Table A.13 represents a 4-exchange in Figure A.3. It represents a 4-exchange which

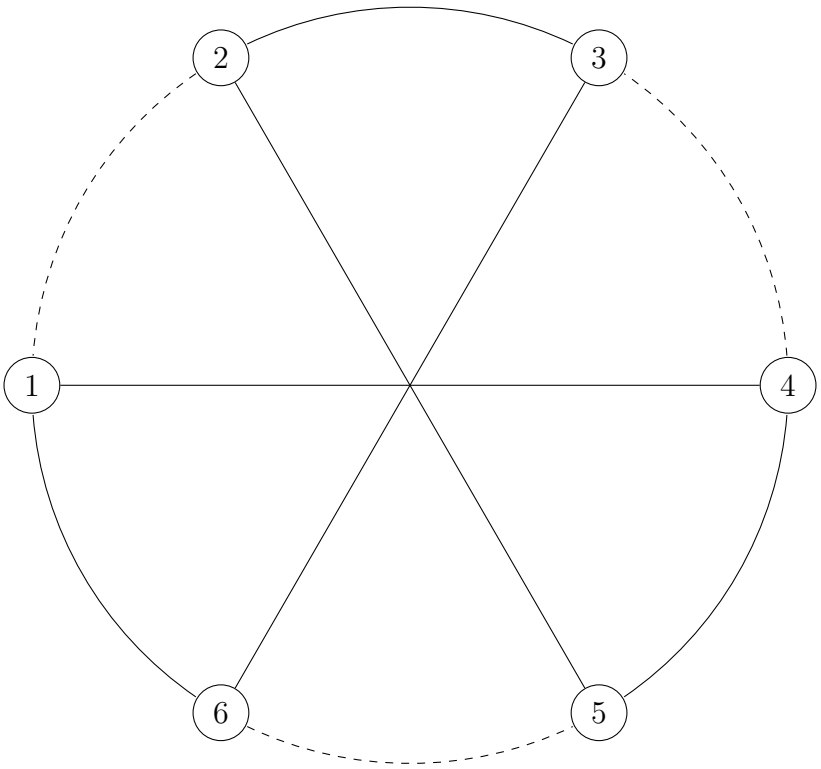


Figure A.2: 3-exchange in TSP ($per3$)

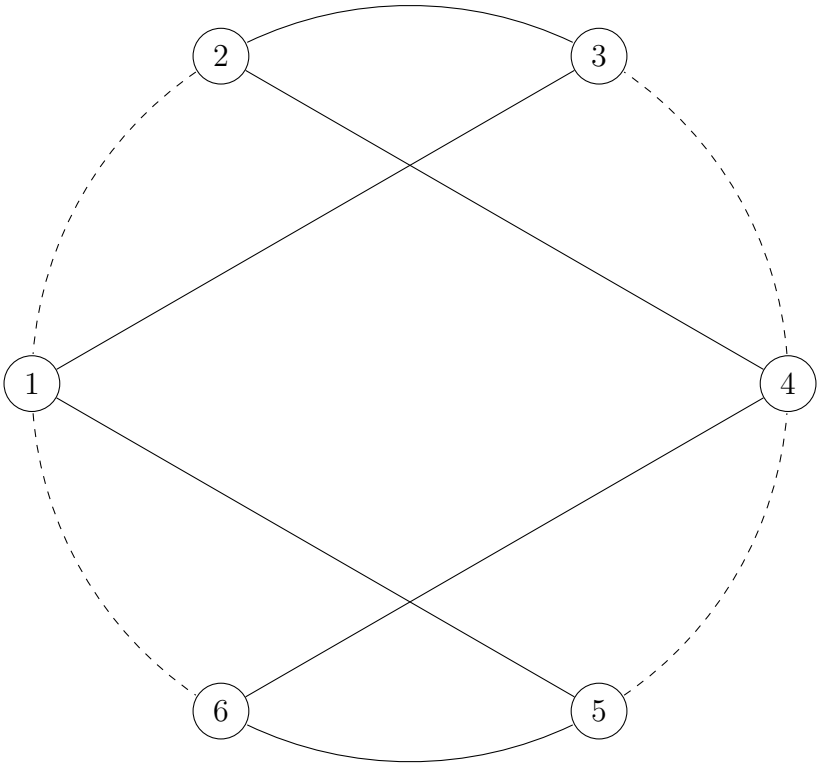


Figure A.3: 4-exchange in TSP ($per4$)

Algorithm A.2: Find a k-exchange in Velocity

Require: Velocity

Start at any row r_0 with nonzero elements.

Let c_0 be a column of an element -1. Let c_1 be a column of an element 1.

Let $X = \{(r_0, c_0)\}$ and $Y = \{(r_0, c_1)\}$

Let $r_1 = c_1$. Go to the row r_1. In the row r_1, find a column c_{-1} which has an element -1.

Let $r_2 = c_{-1}$. Go to the row r_2. In the row r_2, find a column c_1 which has an element 1.

$X = X \bigcup \{(r_1, c_{-1})\}$ and $Y = Y \bigcup \{(r_2, c_1)\}$

Let $r_3 = c_1$. Go to the row r_3. In the row r_3, find a column c_{-1} which has an element -1.

if $(c_{-1}, r_3) = (r_0, c_0)$ **then**

  return X and Y

else

  Let $r_4 = c_{-1}$. Go to the row r_4. In the row r_4, find a column c_1 which has an element 1.

  $X = X \bigcup \{(r_3, c_{-1})\}$ and $Y = Y \bigcup \{(r_4, c_1)\}$

end if

while $true$ **do**

  Let $r_1 = c_1$. Go to the row r_1. In the row r_1, find a column c_{-1} which has an element -1.

  if $(c_{-1}, r_1) = (r_0, c_0)$ **then**

    Break;

  else

    Let $r_2 = c_{-1}$. Go to the row r_2. In the row r_2, find a column c_1 which has an element 1.

    $X = X \bigcup \{(r_1, c_{-1})\}$ and $Y = Y \bigcup \{(r_2, c_1)\}$

  end if

end while

return X and Y

Table A.13: Edge-base Representation of Permutation $per4$

	v1	v2	v3	v4	v5	v6
v1	0	0	1	0	1	0
v2	0	0	1	1	0	0
v3	1	1	0	0	0	0
v4	0	1	0	0	0	1
v5	1	0	0	0	0	1
v6	0	0	0	1	1	0

Table A.14: Velocity $per4 - per1$

	v1	v2	v3	v4	v5	v6
v1	0	-1	1	0	1	-1
v2	-1	0	0	1	0	0
v3	1	0	0	-1	0	0
v4	0	1	-1	0	-1	1
v5	1	0	0	-1	0	0
v6	-1	0	0	1	0	0

is decomposed as a union of two 2-exchanges. The velocity $per4 - per1$ is computed in Table A.14. In general, each nonzero row and each nonzero column in the velocities have (two 1s and two -1s) or (one 1 and one -1). In the rows which have (two 1s and two -1s) we have an option to select an either 1 out of the two 1s and an either -1 out of the two -1s. Therefore the process of finding k-exchanges branches at the rows with (two 1s and two -1s).

In this model, edges among cities (permutations) are represented as incidence matrices. The velocity is computed as the difference of the matrices and decomposed as a union of k-exchanges. Based on these components we can construct a discrete PSO model for TSP. These models are based on the matrix representations which are looked upon as an extension of the real number system.

Bibliography

[1] B. GÄRTNER, M. J., AND MARIA, C. An exponential lower bound on the complexity of regularization paths. *Journal of Computational Geometry 3*, 1 (2012), 168–195.

[2] BALAMURUGAN, P., POSINASETTY, A., AND SHEVADE, S. *ADMM for Training Sparse Structural SVMs with Augmented $\ell 1$ Regularizers*. pp. 684–692.

[3] BANZHAF, W., FRANCONE, F. D., KELLER, R. E., AND NORDIN, P. *Genetic Programming: An Introduction: on the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[4] BAO, Y., HU, Z., AND XIONG, T. A pso and pattern search based memetic algorithm for svms parameters optimization. *Neurocomputing 117* (2013), 98 – 106.

[5] BARTLETT, P., AND SHAWE-TAYLOR, J. Advances in kernel methods. MIT Press, Cambridge, MA, USA, 1999, ch. Generalization Performance of Support Vector Machines and Other Pattern Classifiers, pp. 43–54.

[6] BARTZ-BEIELSTEIN, T., LASARCZYK, C., AND PREUSS, M. The sequential parameter optimization toolbox. In *Experimental Methods for the Analysis of Optimization Algorithms*, T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, Eds. Springer, Berlin, Heidelberg, New York, 2010, pp. 337–360.

[7] BATTITI, R. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks 5*, 4 (July 1994), 537–550.

[8] BAUSCHKE, H. H., AND COMBETTES, P. L. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[9] BECTOR, C.R., B. M., AND KLASSEN, J. Duality for a nonlinear programming problem. *Util. Math. 11* (1977), 87–99.

[10] BERGSTRA, J. S., BARDENET, R., BENGIO, Y., AND KÉGL, B. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 2546–2554.

[11] BERTSEKAS, D. *Nonlinear Programming*. Athena Scientific, 1995.

[12] BHOWAN, U., AND MCCLOSKEY, D. J. *Genetic Programming for Feature Selection and Question-Answer Ranking in IBM Watson*. Springer International Publishing, Cham, 2015, pp. 153–166.

[13] BONABEAU, E., DORIGO, M., AND THERAULAZ, G. *From Natural to Artificial Swarm Intelligence*. Oxford University Press, Inc., New York, NY, USA, 1999.

[14] BONYADI, M. R., AND MICHALEWICZ, Z. A locally convergent rotationally invariant particle swarm optimization algorithm. *Swarm Intelligence 8*, 3 (Sep 2014), 159–198.

[15] BONYADI, M. R., AND MICHALEWICZ, Z. Stability analysis of the particle swarm optimization without stagnation assumption. *IEEE Transactions on Evolutionary Computation 20*, 5 (Oct 2016), 814–819.

[16] BONYADI, M. R., AND MICHALEWICZ, Z. Book review: Particle swarm optimization for single objective continuous space problems: A review. *Evol. Comput. 25*, 1 (Mar. 2017), 1–54.

[17] BOSER, B. E., GUYON, I. M., AND VAPNIK, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (New York, NY, USA, 1992), COLT '92, ACM, pp. 144–152.

[18] BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn. 3*, 1 (Jan. 2011), 1–122.

[19] BRAUN, T. D., SIEGEL, H. J., BECK, N., BÖLÖNI, L. L., MAHESWARAN, M., REUTHER, A. I., ROBERTSON, J. P., THEYS, M. D., YAO, B., HENSGEN, D., AND FREUND, R. F. A comparison of eleven static heuristics for mapping a class of

independent tasks onto heterogeneous distributed computing systems. *J. Parallel Distrib. Comput. 61*, 6 (June 2001), 810–837.

[20] BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery 2*, 2 (Jun 1998), 121–167.

[21] BURJORJEE, K. *SpeedyGA: A Fast Simple Genetic Algorithm Version 1.3*, 2009. Software available at `https://www.mathworks.com/matlabcentral/fileexchange/15164-speedyga-a-fast-simple-genetic-algorithm`.

[22] CAMBINI, A., AND MARTEIN, L. *Generalized Convexity and Optimization*, 1st ed. Springer-Verlag Berlin Heidelberg, 2009.

[23] CHANG, C.-C., AND LIN, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[24] CHAPELLE, O., VAPNIK, V., BOUSQUET, O., AND MUKHERJEE, S. Choosing multiple parameters for support vector machines. *Mach. Learn. 46*, 1-3 (Mar. 2002), 131–159.

[25] CLEGHORN, C. W., AND ENGELBRECHT, A. Fully informed particle swarm optimizer: Convergence analysis. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (May 2015), pp. 164–170.

[26] CLERC, M. *Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 219–239.

[27] CORTES, C., HAFFNER, P., MOHRI, M., BENNETT, K., AND CESA-BIANCHI, N. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research 5* (2004), 1035–1062.

[28] CORTES, C., MOHRI, M., AND ROSTAMIZADEH, A. Learning non-linear combinations of kernels. In *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 396–404.

[29] CORTES, C., AND VAPNIK, V. Support-vector networks. *Mach. Learn. 20*, 3 (Sept. 1995), 273–297.

[30] CRAMMER, K., AND SINGER, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res. 2* (Mar. 2002), 265–292.

[31] DAI, B., XIE, B., HE, N., LIANG, Y., RAJ, A., BALCAN, M., AND SONG, L. Scalable kernel methods via doubly stochastic gradients. *CoRR abs/1407.5599* (2014).

[32] DE JONG, K. A., SPEARS, W. M., AND GORDON, D. F. Using genetic algorithms for concept learning. *Machine Learning 13*, 2 (Nov 1993), 161–188.

[33] DIOSAN, L., ROGOZAN, A., AND PECUCHET, J. P. Evolving kernel functions for svms by genetic programming. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)* (Dec 2007), pp. 19–24.

[34] DIOŞAN, L., ROGOZAN, A., AND PECUCHET, J.-P. Improving classification performance of support vector machine by genetically optimising kernel shape and hyper-parameters. *Applied Intelligence 36*, 2 (2012), 280–294.

[35] DO, H., AND KALOUSIS, A. Convex formulations of radius-margin based Support Vector Machines. In *Proceedings of the 30th International Conference on Machine Learning, Cycle 1* (2013), vol. 28 of *JMLR Proceedings*, JMLR.org, pp. 169–177.

[36] DO, H., KALOUSIS, A., WOZNICA, A., AND HILARIO, M. *Margin and Radius Based Multiple Kernel Learning*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 330–343.

[37] DORIGO, M., AND GAMBARDELLA, L. M. Ant colonies for the travelling salesman problem. *Biosystems 43*, 2 (1997), 73 – 81.

[38] DORIGO, M., MANIEZZO, V., AND COLORNI, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26*, 1 (Feb 1996), 29–41.

[39] DRAKE, J. *Benchmark instances for the Multidimensional Knapsack Problem*, 01 2015. Datasets available at `https://www.researchgate.net/publication/271198281_Benchmark_instances_for_the_Multidimensional_Knapsack_Problem`.

[40] DUBEY, I., AND GUPTA, M. Uniform mutation and spv rule based optimized pso algorithm for tsp problem. In *2017 4th International Conference on Electronics and Communication Systems (ICECS)* (Feb 2017), pp. 168–172.

[41] EBERHART, R. C., AND SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)* (2000), vol. 1, pp. 84–88 vol.1.

[42] EFRON, B., HASTIE, T., JOHNSTONE, I., AND TIBSHIRANI, R. Least angle regression. *Annals of Statistics 32* (2004), 407–499.

[43] ESPEJO, P. G., VENTURA, S., AND HERRERA, F. A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 40*, 2 (March 2010), 121–144.

[44] FAN, R.-E., CHANG, K.-W., HSIEH, C.-J., WANG, X.-R., AND LIN, C.-J. Liblinear: A library for large linear classification. *J. Mach. Learn. Res. 9* (June 2008), 1871–1874.

[45] FARIN, G. *Triangular Bernstein-Bézier patches*. North-Holland, 1986.

[46] FOITHONG, S., PINNGERN, O., AND ATTACHOO, B. Feature subset selection wrapper based on mutual information and rough sets. *Expert Systems with Applications 39*, 1 (2012), 574 – 584.

[47] FRENCH, S. *Sequencing and scheduling: an introduction to the mathematics of the jobshop*. Ellis Horwood series in mathematics and its applications. Ellis Horwood, 1982.

[48] FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software 33*, 1 (2010), 1–22.

[49] GAI, K., CHEN, G., AND ZHANG, C. Learning kernels with radiuses of minimum enclosing balls. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems* (USA, 2010), NIPS'10, Curran Associates Inc., pp. 649–657.

[50] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[51] GIESEN, J., JAGGI, M., AND LAUE, S. Approximating parameterized convex optimization problems. *ACM Trans. Algorithms 9*, 1 (Dec. 2012), 10:1–10:17.

[52] GIESEN, J., LAUE, S., AND WIESCHOLLEK, P. Robust and efficient kernel hyperparameter paths with guarantees. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* (2014), T. Jebara and E. P. Xing, Eds., JMLR Workshop and Conference Proceedings, pp. 1296–1304.

[53] GIESEN, J., MUELLER, J., LAUE, S., AND SWIERCY, S. Approximating concavely parameterized optimization problems. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2105–2113.

[54] GOLDBARG, E. F. G., DE SOUZA, G. R., AND GOLDBARG, M. C. Particle swarm for the traveling salesman problem. In *Evolutionary Computation in Combinatorial Optimization* (Berlin, Heidelberg, 2006), J. Gottlieb and G. R. Raidl, Eds., Springer Berlin Heidelberg, pp. 99–110.

[55] GRAY, H. F., MAXWELL, R. J., MARTNEZ-PREZ, I., ARS, C., AND CERDN, S. Genetic programming for classification and feature selection: analysis of 1h nuclear magnetic resonance spectra from human brain tumour biopsies. *NMR in Biomedicine 11*, 4-5 (1998), 217–224.

[56] GREFENSTETTE, J. J. Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning 3*, 2 (Oct 1988), 225–245.

[57] GRIFFITHS, D. W. *CONTOURING ALGORITHMS WITH TERRAIN MAPPING APPLICATIONS*. PhD thesis, University of Wales, Bangor, School of Mathematics, 2010.

[58] GUYON, I., AND ELISSEEFF, A. An introduction to variable and feature selection. *J. Mach. Learn. Res. 3* (Mar. 2003), 1157–1182.

[59] GUYON, I., WESTON, J., BARNHILL, S., AND VAPNIK, V. Gene selection for cancer classification using support vector machines. *Machine Learning 46*, 1 (Jan 2002), 389–422.

[60] HANSEN, N. *The CMA Evolution Strategy: A Comparing Review*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 75–102.

[61] HARLOW, J., SAINUDIIN, R., AND TUCKER, W. Mapped regular pavings. *Reliable Computing 16* (2012), 252–282.

[62] HASTIE, T., ROSSET, S., TIBSHIRANI, R., AND ZHU, J. The entire regularization path for the support vector machine. *J. Mach. Learn. Res. 5* (Dec. 2004), 1391–1415.

[63] HEIN, M., AND BOUSQUET, O. Kernels, associated structures and generalizations. Tech. Rep. 127, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, July 2004.

[64] HOLLAND, J. H. Outline for a logical theory of adaptive systems. *J. ACM 9*, 3 (July 1962), 297–314.

[65] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. second edition, 1992.

[66] HOWLEY, T., AND MADDEN, M. G. The genetic kernel support vector machine: Description and evaluation. *Artificial Intelligence Review 24*, 3 (2005), 379–395.

[67] IZAKIAN, H., LADANI, B. T., ABRAHAM, A., AND SNÁSEL, V. A discrete particle swarm optimization approach for grid job scheduling. *International Journal of Innovative Computing, Information and Control 6*, 9 (2010), 4219 – 4234.

[68] KAKADE, S. M., SHALEV-SHWARTZ, S., AND TEWARI, A. Regularization techniques for learning with matrices. *J. Mach. Learn. Res. 13* (June 2012), 1865–1890.

[69] KEERTHI, S. S., AND LIN, C.-J. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Comput. 15*, 7 (July 2003), 1667–1689.

[70] KENNEDY, J., AND EBERHART, R. C. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks* (Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995), vol. 4, pp. 1942–1948.

[71] KENNEDY, J., AND EBERHART, R. C. A Discrete Binary Version of the Particle Swarm Algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics* (Washington, DC, USA, October 1997), vol. 5, IEEE Computer Society, pp. 4104–4108.

[72] KENNEDY, J., AND EBERHART, R. C. A discrete binary version of the particle swarm algorithm. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation* (Oct 1997), vol. 5, pp. 4104–4108 vol.5.

[73] KENNEDY, J., AND EBERHART, R. C. *Swarm Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[74] KHANESAR, M. A. *Binary Particle Swarm optimization*, 2013. Software available at `https://au.mathworks.com/matlabcentral/fileexchange/39844-binary-particle-swarm-optimization`.

[75] KHANESAR, M. A., TESHNEHLAB, M., AND SHOOREHDELI, M. A. A novel binary particle swarm optimization. In *2007 Mediterranean Conference on Control Automation* (June 2007), pp. 1–6.

[76] KLOFT, M. *lp-Norm Multiple Kernel Learning*. PhD thesis, Berlin Institute of Technology, 2011.

[77] KLOFT, M., BREFELD, U., SONNENBURG, S., AND ZIEN, A. lp-norm multiple kernel learning. *J. Mach. Learn. Res. 12* (July 2011), 953–997.

[78] KOCH, P., BISCHL, B., FLASCH, O., BARTZ-BEIELSTEIN, T., WEIHS, C., AND KONEN, W. Tuning and evolution of support vector kernels. *Evolutionary Intelligence 5*, 3 (2012), 153–170.

[79] KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2* (San Francisco, CA, USA, 1995), IJCAI'95, Morgan Kaufmann Publishers Inc., pp. 1137–1143.

[80] KONEN, W., KOCH, P., FLASCH, O., BARTZ-BEIELSTEIN, T., FRIESE, M., AND NAUJOKS, B. Tuned data mining: A benchmark study on different tuners. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2011), GECCO '11, ACM, pp. 1995–2002.

[81] KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[82] KOZA, J. R. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines 11*, 3 (Sep 2010), 251–284.

[83] KOZA, J. R. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines 11*, 3-4 (Sept. 2010), 251–284.

[84] KWAK, N., AND CHOI, C.-H. Input feature selection for classification problems. *IEEE Transactions on Neural Networks 13*, 1 (Jan 2002), 143–159.

[85] LANCKRIET, G. R. G., CRISTIANINI, N., BARTLETT, P., GHAOUI, L. E., AND JORDAN, M. I. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res. 5* (Dec. 2004), 27–72.

[86] LARRAANAGA, P., AND LOZANO, J. A. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[87] LASKARI, E. C., PARSOPOULOS, K. E., AND VRAHATIS, M. N. Particle swarm optimization for integer programming. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)* (May 2002), vol. 2, pp. 1582–1587 vol.2.

[88] LI, L., JAMIESON, K. G., DESALVO, G., ROSTAMIZADEH, A., AND TALWALKAR, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research 18* (2017), 185:1–185:52.

[89] LIANG, S., AND SRIKANT, R. Why deep neural networks? *CoRR abs/1610.04161* (2016).

[90] LICHMAN, M. UCI machine learning repository, 2013.

[91] LIN, S., AND KERNIGHAN, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research 21*, 2 (1973), 498–516.

[92] LIN, S.-W., LEE, Z.-J., CHEN, S.-C., AND TSENG, T.-Y. Parameter determination of support vector machine and feature selection using simulated annealing approach. *Appl. Soft Comput. 8*, 4 (Sept. 2008), 1505–1512.

[93] LIU, B., WANG, L., AND JIN, Y. H. An effective pso-based memetic algorithm for flow shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 37*, 1 (Feb 2007), 18–27.

[94] LIU, D., QIAN, H., DAI, G., AND ZHANG, Z. An iterative svm approach to feature selection and classification in high-dimensional datasets. *Pattern Recogn. 46*, 9 (Sept. 2013), 2531–2537.

[95] LIU, G. *Meshfree Methods: Moving Beyond the Finite Element Method, Second Edition*. Taylor & Francis, 2009.

[96] LIU, X., WANG, L., YIN, J., ZHU, E., AND ZHANG, J. An efficient approach to integrating radius information into multiple kernel learning. *IEEE Transactions on Cybernetics 43*, 2 (April 2013), 557–569.

[97] LIU, X., YIN, J., AND LONG, J. *On Radius-Incorporated Multiple Kernel Learning*. Springer International Publishing, Cham, 2014, pp. 227–240.

[98] LU, J., HOI, S. C. H., WANG, J., ZHAO, P., AND LIU, Z.-Y. Large scale online kernel learning. *J. Mach. Learn. Res. 17*, 1 (Jan. 2016), 1613–1655.

[99] LUO, J., ORABONA, F., FORNONI, M., CAPUTO, B., AND CESA-BIANCHI, N. Om-2: An online multi-class multi-kernel learning algorithm.

[100] M., S. B., AND H., S. M. Selection and parameter optimization of svm kernel function for underwater target classification. In *2015 IEEE Underwater Technology (UT)* (Feb 2015), pp. 1–5.

[101] M. A. H. AKHAND, SHAHINA AKTER, M. A. R., AND YAAKOB, S. B. Velocity tentative pso: An optimal velocity implementation based particle swarm optimization to solve traveling salesman problem. *IAENG International Journal of Computer Science 42*, 3 (2015), 221 – 232.

[102] MAK, K.-T., AND MORTON, A. J. Distances between traveling salesman tours. *Discrete Applied Mathematics 58*, 3 (1995), 281 – 291.

[103] MANGASARIAN, O. L. Nonlinear programming, 1969.

[104] MEENAKSHI, A., AND RAJIAN, C. On a product of positive semidefinite matrices. *Linear Algebra and its Applications 295*, 1 (1999), 3 – 6.

[105] MEIROM, E., AND KISILEV, P. Nuc-mkl: A convex approach to non linear multiple kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (Cadiz, Spain, 09–11 May 2016), A. Gretton and C. C. Robert, Eds., vol. 51 of *Proceedings of Machine Learning Research*, PMLR, pp. 610–619.

[106] MENDES, R., KENNEDY, J., AND NEVES, J. The fully informed particle swarm: Simpler, maybe better. *Trans. Evol. Comp 8*, 3 (June 2004), 204–210.

[107] MICCHELLI, C. A., AND PONTIL, M. Learning the kernel function via regularization. *J. Mach. Learn. Res. 6* (Dec. 2005), 1099–1125.

[108] MIRJALILI, S., AND LEWIS, A. S-shaped versus v-shaped transfer functions for binary particle swarm optimization. *Swarm and Evolutionary Computation 9*, Supplement C (2013), 1 – 14.

[109] MITCHELL, M. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[110] MOND, B. Mond-weir duality. In *Optimization*, C. Pearce and E. Hunt, Eds. Springer New York, 2009, pp. 157–165.

[111] MOND, B., AND WEIR, T. Generalized concavity and duality. In *Generalized concavity in optimization and economics*, S. Schaible and W. Ziemba, Eds. Academic Press, 1981, pp. 263–279.

[112] MOSCATO, P. On evolution, search, optimization, genetic algorithms and martial arts towards memetic algorithms.

[113] MOSCATO, P., AND NORMAN, M. G. A "memetic" approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. In *IN PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON PARALLEL COMPUTING AND TRANSPUTER APPLICATIONS* (1992), IOS Press, pp. 177–186.

[114] NESHATIAN, K. *Feature Manipulation with Genetic Programming: A Thesis Submitted to the Victoria University of Wellington in Fulfilment of the Requirements for the Degree of Doctor of Philosophy in Computer Science*. Theses: Computer Science. Victoria University of Wellington, 2010.

[115] NGUYEN, B. H., XUE, B., AND ANDREAE, P. A novel binary particle swarm optimization algorithm and its applications on knapsack and feature selection problems. In *Intelligent and Evolutionary Systems* (Cham, 2017), G. Leu, H. K. Singh, and S. Elsayed, Eds., Springer International Publishing, pp. 319–332.

[116] ORABONA, F. *DOGMA: a MATLAB toolbox for Online Learning*, 2009. Software available at `http://dogma.sourceforge.net`.

[117] ORABONA, F., AND LUO, J. Ultra-fast optimization algorithm for sparse multi kernel learning. Idiap-RR Idiap-RR-11-2011, Idiap, 5 2011.

[118] PARIKH, N., AND BOYD, S. Proximal algorithms. *Found. Trends Optim. 1*, 3 (Jan. 2014), 127–239.

[119] PENG, H., LONG, F., AND DING, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 8 (Aug 2005), 1226–1238.

[120] PLATT, J. C. Sequential minimal optimization: A fast algorithm for training support vector machines. Tech. rep., ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING, 1998.

[121] POLI, R. On the moments of the sampling distribution of particle swarm optimisers. In *Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation* (New York, NY, USA, 2007), GECCO '07, ACM, pp. 2907–2914.

[122] PRAUTZSCH, H., BOEHM, W., AND PALUSZNY, M. *Bezier and B-Spline Techniques*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[123] PUDIL, P., NOVOVIČOVÁ, J., AND KITTLER, J. Floating search methods in feature selection. *Pattern Recogn. Lett. 15*, 11 (Nov. 1994), 1119–1125.

[124] RAHIMI, A., AND RECHT, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds. Curran Associates, Inc., 2008, pp. 1177–1184.

[125] REINGOLD, E. M., NIEVERGELT, J., AND DEO, N. *Combinatorial Algorithms: Theory and Practice*. Prentice Hall College Div, 1977.

[126] RIOLO, R., WORZEL, W. P., KOTANCHEK, M., AND KORDON, A. *Genetic Programming Theory and Practice XIII*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[127] RITCHIE, G., AND LEVINE, J. *A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments*. 12 2004.

[128] ROCKAFELLAR, R. T. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research 1*, 2 (1976), 97–116.

[129] ROCKAFELLAR, R. T. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization 14*, 5 (1976), 877–898.

[130] ROSENBLATT, F. *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957.

[131] SAEYS, Y., INZA, I. N., AND LARRAÑAGA, P. A review of feature selection techniques in bioinformatics. *Bioinformatics 23*, 19 (Sept. 2007), 2507–2517.

[132] SAHA, S., H. M. U. M., AND MONDAL, R. A new approach of solving linear fractional programming problem (lfp) by using computer algorithm. *Open Journal of Optimization 4* (2015), 74–86.

[133] SAHOO, L., BANERJEE, A., BHUNIA, A. K., AND CHATTOPADHYAY, S. An efficient gapso approach for solving mixed-integer nonlinear programming problem in reliability optimization. *Swarm and Evolutionary Computation 19* (2014), 43 – 51.

[134] SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev. 3*, 3 (July 1959), 210–229.

[135] SHALEV-SHWARTZ, S. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, the Hebrew University, 2007.

[136] SHALEV-SHWARTZ, S. Online learning and online convex optimization. *Found. Trends Mach. Learn. 4*, 2 (Feb. 2012), 107–194.

[137] SHALEV-SHWARTZ, S., AND BEN-DAVID, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[138] SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal 27*, 3 (July 1948), 379–423.

[139] SHI, X., LIANG, Y., LEE, H., LU, C., AND WANG, Q. Particle swarm optimization-based algorithms for tsp and generalized tsp. *Information Processing Letters 103*, 5 (2007), 169 – 176.

[140] SHI, Y., AND EBERHART, R. C. A Modified Particle Swarm Optimizer. In *Proceedings of IEEE International Conference on Evolutionary Computation* (Washington, DC, USA, May 1998), IEEE Computer Society, pp. 69–73.

[141] SI, S., HSIEH, C.-J., AND DHILLON, I. S. Memory efficient kernel approximation. *Journal of Machine Learning Research 18*, 20 (2017), 1–32.

[142] SILVA, S., AND ALMEIDA, J. Gplab-a genetic programming toolbox for matlab. In *In Proc. of the Nordic MATLAB Conference (NMC-2003* (2005), pp. 273–278.

[143] SNOEK, J., LAROCHELLE, H., AND ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2951–2959.

[144] SOLIS, F. J., AND WETS, R. J. B. Minimization by random search techniques. *Math. Oper. Res. 6*, 1 (Feb. 1981), 19–30.

[145] SPARKS, E. R., TALWALKAR, A., HAAS, D., FRANKLIN, M. J., JORDAN, M. I., AND KRASKA, T. Automating model search for large scale machine learning. In *Proceedings of the Sixth ACM Symposium on Cloud Computing* (New York, NY, USA, 2015), SoCC '15, ACM, pp. 368–380.

[146] SPEARS, W. M., GREEN, D. T., AND SPEARS, D. F. Biases in particle swarm optimization. *Int. J. Swarm. Intell. Res. 1*, 2 (Apr. 2010), 34–57.

[147] STEARNS, S. D. On selecting features for pattern classifiers. In *Proceedings of the 3rd International Conference on Pattern Recognition (ICPR 1976)* (Coronado, CA, 1976), pp. 71–75.

[148] STEINWART, I., AND CHRISTMANN, A. *Support Vector Machines*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[149] STRASSER, S., GOODMAN, R., SHEPPARD, J., AND BUTCHER, S. A new discrete particle swarm optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (New York, NY, USA, 2016), GECCO '16, ACM, pp. 53–60.

[150] SUGANTHAN, P. N., HANSEN, N., LIANG, J. J., DEB, K., CHEN, Y. P., AUGER, A., AND TIWARI, S. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Tech. rep., Nanyang Technological University, Singapore, 2005.

[151] SULLIVAN, K. M., AND LUKE, S. Evolving kernels for support vector machine classification. In *GECCO '07: Proceedings of the 9th annual conference on Genetic*

and evolutionary computation (London, 7-11 July 2007), D. Thierens, H.-G. Beyer, J. Bongard, J. Branke, J. A. Clark, D. Cliff, C. B. Congdon, K. Deb, B. Doerr, T. Kovacs, S. Kumar, J. F. Miller, J. Moore, F. Neumann, M. Pelikan, R. Poli, K. Sastry, K. O. Stanley, T. Stutzle, R. A. Watson, and I. Wegener, Eds., vol. 2, ACM Press, pp. 1702–1707.

[152] SUREZ, R. R., VALENCIA-RAMREZ, J. M., AND GRAFF, M. Genetic programming as a feature selection algorithm. In *2014 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)* (Nov 2014), pp. 1–5.

[153] TAN, M., TSANG, I. W., AND WANG, L. Towards large-scale and ultrahigh dimensional feature selection via feature generation. *CoRR abs/1209.5260* (2012).

[154] TAX, D. M. J., AND DUIN, R. P. W. Support vector domain description. *Pattern Recognition Letters 20* (1999), 1191–1199.

[155] TIBSHIRANI, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B) 58* (1996), 267–288.

[156] TRAN, B., XUE, B., AND ZHANG, M. Overview of particle swarm optimisation for feature selection in classification. In *Simulated Evolution and Learning* (Cham, 2014), G. Dick, W. N. Browne, P. Whigham, M. Zhang, L. T. Bui, H. Ishibuchi, Y. Jin, X. Li, Y. Shi, P. Singh, K. C. Tan, and K. Tang, Eds., Springer International Publishing, pp. 605–617.

[157] TRAN, B., XUE, B., AND ZHANG, M. A new representation in pso for discretization-based feature selection. *IEEE Transactions on Cybernetics* (2018), 1–14.

[158] TRELEA, I. C. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters 85*, 6 (2003), 317 – 325.

[159] VAN DEN BERGH, F., AND ENGELBRECHT, A. P. A convergence proof for the particle swarm optimiser. *Fundam. Inf. 105*, 4 (Dec. 2010), 341–374.

[160] VAPNIK, V. N. *Statistical Learning Theory*, 1st ed. Wiley, 1998.

[161] VAPNIK, V. N., AND CHERVONENKIS, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications 16*, 2 (1971), 264–280.

[162] VARMA, M., AND BABU, B. R. More generality in efficient multiple kernel learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY, USA, 2009), ICML '09, ACM, pp. 1065–1072.

[163] VERMA, R. S., AND KUMAR, S. Dsapso: Dna sequence assembly using continuous particle swarm optimization with smallest position value rule. In *2012 1st International Conference on Recent Advances in Information Technology (RAIT)* (March 2012), pp. 410–415.

[164] VIKHAR, P. A. Evolutionary algorithms: A critical review and its future prospects. In *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)* (Dec 2016), pp. 261–265.

[165] WEI-CHENG CHANG, C.-P. L., AND LIN, C.-J. A revisit to support vector data description (svdd). Tech. rep., 2013.

[166] WENDLAND, H. *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004.

[167] WIENER, N. *Cybernetics, or, Control and communication in the animal and the machine / Norbert Wiener*. Technology Press [Cambridge, Mass.], 1948.

[168] WILLIAMS, C. K. I., AND SEEGER, M. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2001, pp. 682–688.

[169] XUE, B., ZHANG, M., BROWNE, W., AND YAO, X. A survey on evolutionary computation approaches to feature selection. *Evolutionary Computation, IEEE Transactions on PP*, 99 (2015), 1–1.

[170] XUE, B., ZHANG, M., AND BROWNE, W. N. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Transactions on Cybernetics 43*, 6 (Dec 2013), 1656–1671.

[171] YAMADA, S., AND NESHATIAN, K. Hyper-parameter search in support vector machines using PSO with cellular fitness approximation. In *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017, Honolulu, HI, USA, November 27 - Dec. 1, 2017* (2017), pp. 1–8.

[172] YAMADA, S., AND NESHATIAN, K. Kernel construction and feature subset selection in support vector machines. In *Simulated Evolution and Learning - 11th*

International Conference, SEAL 2017, Shenzhen, China, November 10-13, 2017, Proceedings (2017), pp. 605–616.

[173] YAMADA, S., AND NESHATIAN, K. Multiple kernel learning with one-level optimization of radius and margin. In *AI 2017: Advances in Artificial Intelligence - 30th Australasian Joint Conference, Melbourne, VIC, Australia, August 19-20, 2017, Proceedings* (2017), pp. 52–63.

[174] YAMADA, S., NESHATIAN, K., AND SAINUDIIN, R. Optimal hyper-parameter search in support vector machines using bézier surfaces. In *AI 2015: Advances in Artificial Intelligence - 28th Australasian Joint Conference, Canberra, ACT, Australia, November 30 - December 4, 2015, Proceedings* (2015), B. Pfahringer and J. Renz, Eds., vol. 9457 of *Lecture Notes in Computer Science*, Springer, pp. 623–629.

[175] YANG, Q., FU, H., AND ZHU, T. An optimization method for parameters of svm in network intrusion detection system. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)* (May 2016), pp. 136–142.

[176] YANG, X.-S. *A New Metaheuristic Bat-Inspired Algorithm*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 65–74.

[177] YU, L., AND LIU, H. Efficient feature selection via analysis of relevance and redundancy. *J. Mach. Learn. Res. 5* (Dec. 2004), 1205–1224.

[178] YUAN, G.-X., CHANG, K.-W., HSIEH, C.-J., AND LIN, C.-J. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *J. Mach. Learn. Res. 11* (Dec. 2010), 3183–3234.

[179] YUKAI YAO, HONGMEI CUI, Y. L. L. L. L. Z., AND CHEN, X. Pmsvm: An optimized support vector machine classification algorithm based on pca and multilevel grid search methods. *Mathematical Problems in Engineering 2015* (2015).

[180] ZHEN, L., WANG, L., HUANG, Z., AND ZHEN, L. Probability-based binary particle swarm optimization algorithm and its application to wfgd control. *2008 International Conference on Computer Science and Software Engineering (CSSE 2008) 01* (2008), 443–446.

[181] ZHEN, L., WANG, L., WANG, X., AND HUANG, Z. A novel pso-inspired probability-based binary optimization algorithm. In *2008 International Symposium on Information Science and Engineering* (Dec 2008), vol. 2, pp. 248–251.

[182] ZHIGANG YAN, Y. Y., AND DING, Y. An experimental study of the hyper-parameters distribution region and its optimization method for support vector machine with gaussian kernel. *International Journal of Signal Processing, Image Processing and Pattern Recognition 6*, 5 (2013), 437–446.